DEPARTMENT OF DEFENCE
DEFENCE SCIENCE & TECHNOLOGY ORGANISATION
DSTO

Integrated Visualisation and Description of Complex Systems

D.P.J. Goodburn, R.J. Vernik,
M.P. Phillips and J.J. Sabine

DSTO-RR-0154

Integrated Visualisation and Description of Complex Systems

D.P.J. Goodburn, R.J. Vernik, M.P. Phillips and J.J. Sabine

Information Technology Division

Electronics and Surveillance Research Laboratory

DSTO-RR-0154

ABSTRACT

This report discusses the Integrated Visualisation and Description (IV&D) Approach, a computer-based visualisation approach that is being developed to support the visualisation and description of complex systems. Guided by a conceptual model of a description process that is driven by user information needs within a domain context, the approach incorporates the use of novel visualisation techniques based on system topographies and feature overlays. System information from the domain's information space is filtered and integrated into a Composite Systems Model that provides a basis for consistency and integration between all system views. A component-based software engineering methodology is described for the development of domain assets which provide the basis for developing a family of domain specific visualisation tools, called Integrated Visualisation Environments (IVEs), and deploying IVEs in their domain of use through the support of software agents. Examples of use of the IV&D Approach for the visualisation of heterogeneous software systems, C2* systems-of-systems and enterprise architectures are discussed, as are future research issues to be investigated.

RELEASE LIMITATION

Approved for public release

DEPARTMENT OF DEFENCE
DEFENCE SCIENCE & TECHNOLOGY ORGANISATION
DSTO

Published by

DSTO Electronics and Surveillance Research Laboratory
PO Box 1500
Salisbury South Australia 5108 Australia

Telephone: (08) 8259 5555
Fax: (08) 8259 6567

AR-010-993
June 1999

APPROVED FOR PUBLIC RELEASE

Integrated Visualisation and Description of Complex Systems

Executive Summary

Understanding systems is important to their design, development, management, use and maintenance. An important step towards understanding systems is gaining the ability to visualise them. However, the ease with which one can visualise a system often depends on its complexity. While system complexity is a function of the number of elements and relationships that constitute a system, it also relates to other properties of systems such as the types of elements and relationships making up the system (esp. whether they are concrete or conceptual), the evolving nature of the system, predictability and whether it is directly visible.

In the Defence domain, as with many domains, the advances in and use of information technology have brought about an increase in system complexity. For example, the advent of Command, Control, Communications and Intelligence (C3I) systems has resulted in the combination of already complex Defence systems such as ground, naval, air, over-the-horizon radar, sensor systems, computer and communications systems into a new, higher-order class of system, which we refer to in this report as the C2* system-of-systems. The characteristic evolutionary and emergent behaviours of these higher-order systems further add to their complexity.

Computer-based visualisation is one method that has been used to aid in system understanding. However, the effectiveness of current computer-based system visualisation methods has been limited by problems relating to the complexity of the system being visualised such as scalability and scope. It is important to develop new ways of visualising and describing systems that deal with complexity and focus on the needs of the user by providing greater levels of flexibility and user support.

The aim of the work described in this report is to provide a means to model, describe and visualise complex systems in order to support the work processes of those involved with planning, developing, and enhancing the overall capability of the systems. For example, in the C2* domain this might include the support of capability planners, developers, project managers and those involved with assessing particular properties of the system (e.g. vulnerability analysis). The report highlights problems in using information to describe particular aspects of systems and the limitations of current computer-based visualisation tools in providing support in the visualisation of systems.

A new approach to the problem of describing large complex systems through the use of computer-based integrated visualisations is proposed. This new approach is called the Integrated Visualisation and Description Approach. The key features of the IV&D Approach are:

- A conceptual model of the system description process that focuses on supporting the user through facilitation, mediation and provision subprocesses.
- The use of novel topographic views in conjunction with more familiar supporting views for the description of systems.
- A Composite Systems Model that integrates system information filtered automatically from the domain's information space and provides a basis for consistency and integration between all system views.
- The use of a component-based software engineering methodology to develop domain assets which in turn provide the basis for assembling a family of domain specific visualisation tools called Integrated Visualisation Environments (IVEs).
- The use of software agents as a support mechanism for the deployment of IVEs into their domain of use.

Examples are given of the application of the IV&D Approach for the visualisation of heterogeneous software systems, C2* systems-of-systems and enterprise architectures. The report concludes by discussing future research issues to be investigated relating to the development and application of the IV&D Approach.

Authors

Daniel Goodburn
Information Technology Division

Dr. Daniel Goodburn is employed as a Research Scientist in the Software Systems Engineering Group. His current research interests are software agents and systems visualisation. Previous work includes research into the use of expert systems to support mechanical engineering designers. He has a Bachelor of Science in the Faculty of Mathematical and Computer Science (Honours) from the University of Adelaide and a Ph.D. in Information Technology from the University of South Australia.

Rudi Vernik
Information Technology Division

Dr. Rudi Vernik is employed as a Principal Research Scientist and is Head of Software Systems Engineering Group. His research interests focus on the definition, development, and application of new systems and software engineering approaches to support Defence in its development of capabilities that will facilitate knowledge and information-based warfare. His group is currently undertaking research in the areas of systems visualisation, component-based software engineering, systems characterisation and modelling, systems dynamics, and evolutionary capability development processes. Rudi has a Ph.D. in Computer and Information Science (Software Engineering) from the University of South Australia. He also has a Bachelor of Electronics Engineering (with Distinction) and a Diploma of Communications Engineering from the Royal Melbourne Institute of Technology.

Matthew Phillips
Information Technology Division

Matthew Phillips is a researcher employed in Software Systems Engineering Group, Information Technology Division. His research interests include distributed systems, programming language design, software visualisation and object-oriented software engineering. Matthew has a Bachelor of Computer Science (with Honours) from The University of Adelaide.

Justin Sabine
Information Technology Division

Justin Sabine is a researcher employed in Software Systems Engineering Group, Information Technology Division. His research interests include interoperability issues, software visualisation, object-oriented software engineering, and agent technologies. Justin has a Bachelor of Arts degree with majors in Computer Science and Psychology, and a Bachelor of Science (Honours) in Computer Science from Flinders University.

Contents

Abbreviations

ACSS	Air Command Support System
ADF	Australian Defence Force
ADO	Australian Defence Organisation
BCSS	Battlespace Command Support System
C2(*)	Command and Control (and other integrated C2 support systems)
C3I	Command Control Communications and Intelligence
C4ISR	Command Control Communication Computers Intelligence Surveillance and Reconnaissance
CBSE	Component Based Software Engineering
CSM	Composite Systems Model
CSS	Command Support System
DEFSECNET	Defence Security Network
EXC3ITE	Experimental C3I Technology Environment
IV&D	Integrated Visualisation and Description
IVD	Integrated Visual Description
IRM	Information Resource Model
IVE	Integrated Visualisation Environment
JCSE	Joint Command Support Environment
SOS	System(s)-of-Systems
US DoD	United States Department of Defence
VIDECS	Visualisation and Description of Complex Systems

1. Introduction

Understanding systems is important to their design, development, management, use and maintenance. An important step towards understanding systems is gaining the ability to visualise them. However, the ease with which one can visualise a system often depends on its complexity. While system complexity is a function of the number of elements and relationships that constitute a system, it also relates to other properties of systems such as the types of elements and relationships making up the system (esp. whether they are concrete or conceptual), the evolving nature of the system, predictability and whether it is directly visible.

In the Defence domain, as with many domains, the advances in and use of information technology have brought about an increase in system complexity. For example, the advent of Command, Control, Communications and Intelligence (C3I) systems has resulted in the combination of already complex Defence systems such as ground, naval, air, over-the-horizon radar, sensor systems, computer and communications systems into a new, higher-order class of system. This new class of systems, called systems-of-systems, is characterised by the independence of system components, the evolutionary nature of the system, emergent behaviour, and a geographic extent that limits the interaction of system components to information exchange (Maier 1997). These characteristics, particularly the evolutionary and emergent behaviours add further dimensions to the complexity of these systems. In this report we refer to the overall system of systems that support command and control as the C2* system-of-systems, where the "*" refers to the range of systems that support command and control.

Computer-based visualisation is one method that has been used to aid in system understanding. However, the effectiveness of current computer-based system visualisation methods has been limited by problems relating to the complexity of the system being visualised, such as scalability and scope. It is important to develop new ways of visualising and describing systems that deal with complexity and focus on the needs of the user by providing greater levels of flexibility and user support.

The aim of this report is to provide an overview of the key concepts involved in a new approach to the visualisation of large complex systems – the Integrated Visualisation and Description (IV&D) Approach. The report illustrates how this approach might be applied in practice by giving examples of its application to three system visualisation problems and discusses research issues to be investigated and case studies to be conducted in future work.

Section 2 begins with a definition of the term 'system' and discusses the need for effective system descriptions. Section 3 examines computer-based visualisation as a method for the description of systems and discusses limitations of current techniques. This leads into section 4 which discusses the IV&D Approach. The key elements of this approach are elaborated, including the conceptual model of the description process, integrated visual descriptions, the Composite Systems Model, Integrated Visualisation

Environments and the use of agent-based support tools. Section 5 gives details of scenarios of use of the IV&D Approach for the visualisation of three types of system – a heterogeneous software system, a C2* system-of-systems, and an enterprise architecture.

Section 6 gives details of future research issues to be investigated and a summary and conclusions are made in Section 7.

2. The Need for Effective Systems Descriptions

In this section, the concept of a system is reviewed and several definitions of 'system' are presented. The recent focus by industry and Defence on the development of increasingly complex systems, including systems-of-systems, is discussed to highlight the importance of developing methods that promote understanding of complex systems. The concept of system descriptions is discussed – their role within the context of systems understanding and the problems that need to be addressed in order to create effective system descriptions.

The section concludes with an account of the authors' experiences in the description of large software systems, the problems encountered, and the lessons learned. Other systems are then considered and it is argued that the concepts and techniques developed to solve the problems of description in the software domain should also apply in other system domains.

2.1 What is a system?

A system, as defined in the Macquarie Dictionary, is an assemblage or combination of things or parts forming a complex or unitary whole: *a mountain system, a railway system.* The NASA Systems Engineering Handbook (NASA 1992) provides a definition of a system as "a set of interrelated components which interact with one another in an organised fashion toward a common purpose." The former definition identifies the duality of systems, in that they can be thought of as a single entity or a set of elements. The latter definition highlights the importance of relationships between component elements within a system. The US DoD definition of system is "any organised assembly of resources and procedures united and regulated by interaction or interdependence to accomplish a set of specific functions" (US DoD 1998). It is important to note that the components that make up a system need not only be hardware or software, but can include people and processes that contribute to the function of the system.

Kaposi and Pyle (1993) also note that a system can be considered as both a unity (a single entity) and a composition (made up of many parts). When being considered as a unity, a system may be characterised by its various properties or attributes and the relationships between these attributes. For instance, a car may be characterised by attributes such as cost, style, seating capacity, colour, maximum speed, comfort, etc. and these attributes may be related in some way (e.g. the greater the seating capacity the greater the cost). As a composition, a system may be characterised as a structure

made up from other elements with a set of relationships between the elements. For example, a car is a structure that is made up from wheels, axles, brakes, engine, chassis, doors, windows etc. with various relationships existing between these elements such as wheels are connected to the axles. Elements that make up a system may also be considered to be systems in their own right. A formal definition of a system will be used later on in section 4.4 to define the concept of a Composite Systems Model.

The types of systems that interest us in this report are complex information systems. System complexity may be characterised by the number of elements that make up a system, the variety of elements, the number of relations between system elements, and the number of attributes and attribute types that describe each system element or relationship. How these elements, attributes and relationships are arranged within the system is also important (i.e. the level of interconnectivity, the number of patterns within the system). In general, the larger the value of each of these characteristic variables, the greater the complexity. Complexity increases further when the system is dynamic, with the elements and relationships between elements evolving over time, and also if the system is chaotic, with some level of uncertainty pertaining to the relations between elements and the attributes of the elements themselves. An example of a complex system is a software system, such as a C3I system used by Defence or a banking system used by a financial institution. Such software systems may have existed for a long period of time, and have grown and evolved during this time to encompass many software components written in different languages, at different times, with many source lines of code. Components may also exist on different platforms distributed over a large network with large numbers of both physical and logical relationships existing between the different modules of the system.

2.2 Systems-of-systems

Recently there has been interest in the Defence domain in "systems-of-systems" (SOS) (Allison and Cook 1997; Maier 1997; Owens 1997). Maier (1997) distinguishes systems-of-systems from large but monolithic systems by "the independence of their components, their evolutionary nature, emergent behaviours, and a geographic extent that limits the interaction of their components to information exchange." These systems-of-systems can either be organised and managed to express particular functions or can be those in which desired behaviours must emerge through voluntary and collaborative interaction (Maier 1997). Examples of systems-of-systems given by Maier (1997) are integrated air-defence systems, the Internet, intelligent transport systems, economic and social systems.

The domain of Command, Control, Communication and Intelligence (C3I) is another where the system-of-systems concept has become increasingly important. The purpose of a C3I system is to provide commanders with tools designed to capture raw data, turn it into useable information and then make it available as knowledge to the right person, at the right time, in the right place and in the right format (DSTO 1997). As such, C3I systems consist of a plethora of systems such as communications and information systems, some of which may be systems-of-systems in their own right,

each designed to capture and/or process data with a view to fulfilling this need. C3I is itself often seen as a system within a larger system. For example, the notion of C4ISR includes elements of computers, surveillance and reconnaissance. This in turn plays a part in the overall national Defence system. Rather than continually creating new acronyms to indicate the integration of systems that comprise the Defence command and control capability, these Command and Control (C2) systems-of-systems will be referred to as C2* systems, where the "*" refers to the range of systems that support command and control (e.g. communications, personnel, logistics, etc.).

The development of these new systems-of-systems heightens the need for new ways of coping with system complexity to promote understanding, development and maintenance of systems. An understanding of a SOS requires some understanding of each system making up the SOS and how each system contributes to the behaviour of the SOS. When one considers that each constituent system may be highly complex or even a SOS itself this is a non-trivial task. Indeed Allison and Cook (1997) claim that "we lack ... knowledge of the underlying generators of total system behaviour for the particular class or classes of complex systems we have to deal with in defence." In addition to their structural complexity, SOSs are dynamic and evolve over time. The World Wide Web is an example of a SOS that is continually growing and evolving without any controlling body. Maier (1997) provides another extreme example of the evolutionary nature of an integrated air defence system that may have its component parts forcibly removed (i.e. destroyed) during an attack. The evolutionary nature of SOSs adds another dimension to their complexity and further contributes to the problem of understanding the system. The next section looks at the problem of understanding systems in terms of system description.

2.3 Describing systems

Large complex systems can be difficult to visualise and hence understand. For example, in the C2* domain, systems comprise of a variety of distributed computer hardware components, communications networks, software, procedures, data, and people. Although some elements of the system are directly visible (e.g. the hardware components), many of the key aspects of the system are intangible and conceptual in nature (e.g. software, information flows, interactions etc). Although some key aspects are not directly visible through observation, information about the system can be used to describe and hence help us visualise these aspects of interest. Visualisation refers to the ability to "make visual or visible" to the eye (Delbridge, Bernard et al. 1991). The dictionary also refers to the fact that visualisation "make[s] perceptible to the mind" (Delbridge, Bernard et al. 1991). This is important since visualisation needs to support an individual's conceptualisation (e.g. mental model) of the system. In many instances, there needs to be a common conceptualisation of the system to support tasks in which several individuals might be involved.

A description is information that represents something of interest. Descriptions provide a form of indirect visibility and hence help us to visualise. Vernik (1996) suggests that descriptive information (i.e. descriptions) has the following

characteristics: *it is human usable, it refers to something, it must provide a valid representation of the thing of interest, and it must be used with appropriate contextual knowledge*. Although descriptive information is commonly considered in terms of media that makes use of visual senses (i.e visual descriptions), other forms of description can be used such as sound. This report focuses on the use of visual descriptions of systems.

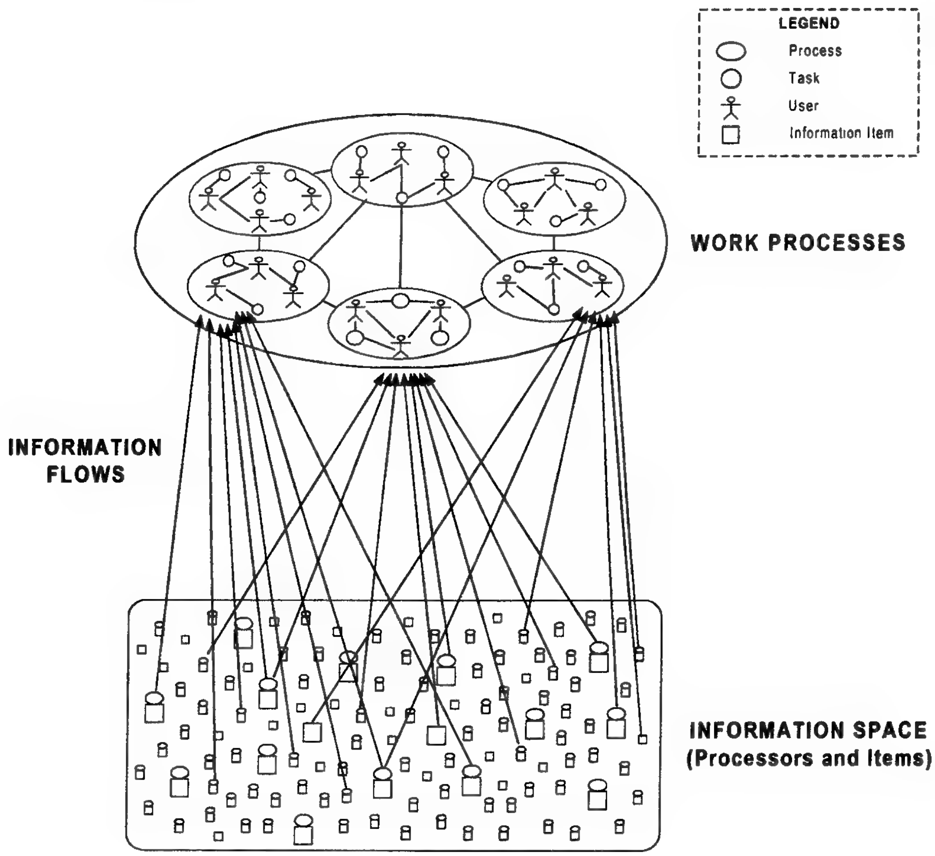

Figure 2-1 – Generalised context of the system description problem

Figure 2-1 illustrates the generalised context of the system description problem. The ellipses seen in the top of the figure represent a set of work processes relating to some generic system. Examples of such processes might be standard lifecycle processes such as acquisition, analysis, synthesis, configuration management, quality assurance, and evolution. Each of these processes requires people to perform a variety of tasks (indicated by circles). In order to perform these tasks, the people rely on information gathered about the system. Each individual user may have different information needs resulting in complex information flows from the information space.

The information space illustrated in Figure 2-1 is considered to be the collection of all information sources and items that contain information that describes some aspect of

the system. Information items (indicated by squares in the figure) could come in the form of documents, models within design environments, metrics reports, or some other form of information. Information items may include a person's knowledge about the system and may be explicit or tacit. Associated with many of these items are information processors that produce and provide the information items. Information processors (indicated by ellipses in the information space) may be human, or non-human, such as a software measurement tool that automatically generates a set of measures that describe system properties, or a CAD system that produces a bill of parts. The common characteristic of all information items produced is that they all describe some aspect of the system. Each information item therefore contributes to a description of the system as a whole. However, this does not mean that the combination of all the information items represents an effective description of the system.

What is meant by an effective system description? The effectiveness of a description relates primarily to the purpose of the description and how well it achieves its purpose. One purpose of a system description is to communicate information to users so they may better understand the system. Description may also be used as a basis for comparison between one system and another or between different states of the same system. An effective description shields the user from the complexity of the system. It is able to extract relevant descriptive information from information items and combine various types of information from one or several types of information items into a consistent form. It can adapt to the changing needs of users whether those changes relate to the task being performed by the user or the context in which the task is being performed. It can adapt to different users. An effective system description must also reflect the dynamic properties of systems. It must communicate changes in system information in a timely manner and have the ability to present the temporal properties of the system. The system description must allow the user to visualise and understand the system in a way that they can make accurate and timely decisions. Ultimately, the judgement of a description's effectiveness will depend on how well the description helps the user to understand the system and so is linked to the measure of the user's understanding. Evaluating the effectiveness of a system description is a research issue of considerable import that has not yet been fully addressed.

2.4 Description of Software and Other Systems

DSTO has had a longstanding research programme which has focused on providing more effective descriptions for the acquisition, development and maintenance of Defence software systems. This work has covered a variety of areas such as documentation approaches, the use of metrics, Computer Aided Software Engineering (CASE) tools, and computer-based visualisation techniques.

The research identified a range of issues and problems with many forms of systems information used to support individuals involved with software related activities. These are summarised in Table 2.1 and discussed in depth in Vernik (1996). For example, copious amounts of documentation was (and still is) produced by software

projects. However, individuals found it difficult to identify and access the specific information they required. Moreover, since the documentation was difficult to maintain, it was often out of date and so did not accurately describe those aspects of interest. Metrics information provided other challenges. The amount of information produced by metrics tools was overwhelming. For example, one of the systems studied comprised 739 source code files for which the metric tool produced some 162,580 measures. This amounted to some 2,217 pages of metric information. Although some of the metrics proved to be good descriptors of particular software characteristics, this information often needed to be used with other descriptive information if it was to be used effectively. For example, the product metric information needed to be used together with information that described code structure (e.g. from documents or CASE tools) to highlight particular areas of the system where quality problems prevailed. Integration with other information and representations such as process-based attributes (e.g. degree of testing, degree of change, defects) and resource information (e.g. the author of particular sections of code) also proved to be important.

PROBLEMS
User is overwhelmed by the **quantity** of information.
Representations do not **scale**.
Information is not **available** at the appropriate time.
Superfluous and **redundant** information is provided.
Information is misleading. It does not **communicate** the appropriate message.
Information is **costly** to provide, use, and maintain.
Information is difficult to **access**.
Information from more than one source is difficult to **assimilate**.
Information is not **consistent** over time or between representations.
Information does not effectively **describe** the aspects of interest.

Table 2.1 - Problems associated with describing systems

The use of computer-based visualisation approaches was seen as a possible solution to many of the problems. Initial research looked at integrating product metrics information with structural representations to help describe particular features of the software such as descriptiveness, complexity, and programming practices. An initial visualisation environment (SEE-Ada Version 1) was developed to support this phase of

the research. Later research phases expanded the visualisation concepts to allow the integrated visualisation of a range of project and product information including system configuration, requirements mappings, design details, test data, and defect data. The approach was used in the field to support a variety of users and tasks ranging from project management to software maintenance. A substantial industry-based case study was undertaken, specifically to study the effects that these types of integrated visualisation approaches would have on particular roles such as software team leaders. This involved characterising the application domain in terms of underlying information sources and the descriptive information needs of individuals. An instrumented visualisation environment and associated automation tools were then deployed and monitored. The results of this work helped validate many of the underlying concepts discussed in this report.

In demonstrating the concepts to individuals in other domains, it became evident that the visualisation approaches that we had defined could be extended and applied more generally to other system description problems. For example, many complex systems exhibit the same underlying characteristics as software systems – they consist of large numbers of different types of entities (often conceptual in nature), there are many complex relationships between the entities that need to be described, and they have no underlying physical form or shape. Examples include the World Wide Web, C2* systems of systems, systems of military strategy, and enterprise architectures. This report draws from our experiences in applying computer-based visualisation approaches for the description of software systems and considers how these techniques could be applied more generally to support the description of other complex systems.

3. Use of Computer-Based Visualisation

This section discusses the use of computer-based visualisation as a basis for system description. The benefits of using computer-based visualisation are examined, in particular its support of exploration within problem domains. Several examples of computer-based tools that support visualisation are discussed. The chapter concludes by focusing on the limitations of current visualisation tools and highlights features required to enhance the process of system description.

3.1 What is Computer-Based Visualisation?

McCormick, DeFanti et al. (1987) define computer-based visualisation as the "study of mechanisms in computers and in humans which allow them in concert to perceive, use, and communicate visual information". This definition implicitly captures several important points. Computer-based visualisation is not just about producing visual representations of information. The information produced has a particular purpose – it is descriptive, in the sense characterised in Section 2.3. A computer-based visualisation can provide a description that conveys information to a user about something. Computer-based visualisation also encapsulates the idea of how visual information is used and communicated. It is important not only to concentrate on mechanisms of

human perception but to also study the tasks for which users require visual information and how the visual information supports those tasks. Finally, McCormick, DeFanti et al. (1987) have suggested that computers and humans work "in concert". The interaction between computers and human users is an important aspect of computer-based visualisation.

This last point is particularly pertinent in the context of using computer-based visualisation as a basis for the description of complex systems. The large and dynamic nature of these systems means that the visualisations that describe them may be complicated and consist of a number of views showing large numbers of system elements and relationships that change over time. Similarly, users' information requirements will change over time as the system evolves and the users' understanding of the system develops. A flexible method of interaction between users and the computer is necessary to customise and adapt visualisations to users' specific and current information needs.

3.2 Potential Benefits of Computer-Based Visualisation

Computer-based images are becoming a routine way of conveying information for a range of tasks in a variety of application areas. For example, they are being used to help support a range of scientific research endeavours through scientific visualisation (Brodlie, Carpenter et al. 1992), to solve crimes (Davidson 1993), and to support the diagnosis of medical conditions (Rogers 1995). Computer-based visualisation is also used in more common everyday circumstances, for instance to convey weather information on the news or during sport commentary for comparison of performances of opposing teams.

Computers are useful tools for providing visualisations of information because of the variety of visualisation techniques that can be employed using computers and the flexibility and speed at which visualisations can be generated and adapted. The use of graphical, direct manipulation interfaces (Shneiderman 1983) allows users to interact with the visualisations presented on the computer screen. The visualisations can be tailored to suit particular users and changes made are immediately reflected on the screen. Computer-based methods of description have further advantages such as the ability to use simulation and animation. Dynamic information can be illustrated using computer animation to show how computer-based descriptions evolve over time. Simulation can be used to predict how systems being visualised will behave under certain conditions, and several such visualisations may then be displayed concurrently for comparison. Finally, computers are often used for the storage of information, and information in this form is then easily accessible to computer-based visualisation tools.

3.3 Visualisation Approaches

3.3.1 General Uses of Visualisation

There are a vast number of approaches and tools that provide computer-based visualisations in a large number of recreational and vocational domains. Various types of charts are used by cricket commentators when discussing the performances of cricket teams or individuals. One-day cricket fans would be very familiar with the "worms" and graphs that are used to compare run-rates between opposing teams and to indicate fall-of-wickets, and to judge the form of their favourite cricketers. Many video games use visualisation to convey game information to the player (e.g. maps in strategy games, gauges showing levels of health, ammunition or fuel in action games). Satellite imagery for weather forecasting and illustration of current rainfall data, and charts showing UV readings and pollen counts are commonplace on television news programmes.

In the field of medicine, medical imaging is used to identify abnormalities and diagnose illness. Rogers (1995) discusses the use of computer-based visualisation of chest X-rays to support the diagnosis of diseases. Likewise magnetic resonance imaging is used to create 2D cross-section images of the brain used for diagnosis. 3D animation is used for simulation of aircraft and car accidents in order to study the impact on the vehicles and occupants (USAF 1998) or as evidence in law courts (Jeffrey 1995). Computer-aided design systems also use animation to check the behaviour of mechanical systems. Finite element methods provide the basis for other visualisation techniques for analysis of design artifacts to study, for example, the effect of mechanical stress or temperature (LUSAS 1998). *AT&T Interactive Data Visualization* (AT&T 1995) is a suite of visualisation tools from AT&T Labs that provides a variety of novel visualisations for viewing data in domains such as telecommunications, finance, and manufacturing.

Many of the example visualisation techniques described above have the advantage that the entities being visualised have an underlying physical form that sets a context for overlaying attribute information of the systems. However, for many synthetic systems, for example, software systems and intranets, there is no such underlying physical form to provide a context for viewing system information, and developers of visualisation techniques must impose some representation of system structure. Typically, the structure of these types of systems is represented in terms of a directed graph and there have been attempts to integrate data visualisation with these structural representations (Imagix 1998). However, for large and complex systems, directed graphs suffer from problems of scalability and lack a compact spatial form.

3.3.2 Visualisation of Software Systems

In the software domain, visualisation is often used to convey information about software code modules and relationships between modules. For example, *Imagix 4D* from Imagix Corporation is a commercially available software visualisation tool that

provides three-dimensional visualisation of C and C++ software (Imagix 1998). The *SeeSoft* system (Eick, Steffen et al. 1992) uses a reduced representation of source code files for providing a range of code-related information such as code history, change details and defects. *Rational Rose* (Rational Software Corporation 1998) provides a number of software design views based on the standard UML diagram types. These include class diagram views showing class interfaces and their static relationships to each other, sequence and interaction diagrams showing the dynamic behaviour of classes, and component diagrams showing how design elements are mapped to physical components. Duffett and Vernik (1997) give an account of the Netmap database visualisation and analysis tool and its application to the visualisation of large software systems. In this account they highlight useful features of the Netmap tool such as the capability to gain an overview (or "footprint") of the system being visualised, the ability to quickly move between different system displays and the ability to hide irrelevant information.

The *SEE-Ada* system (Vernik 1996) is a tool that supports the visualisation of large Ada software systems by combining information from a range of project sources and integrating this information in a consistent manner into a number of primary views of the software system. The primary views are a type of software map which can be customised to provide integrated information for a variety of perspectives and needs. These primary views are supported through the provision of peripheral views that provide alternative views of the information shown in the primary views and additional information required by the user. All views are customisable and tailorable by the user to suit their particular information needs. A key feature of this approach is that the views are generated from an underlying model of the software. This model provides a means of integrating information and ensures that all views are consistent.

3.3.3 Visualisation of Systems-of-Systems

Although there has been some work in visualising large systems, much of this work has focused on visualisation of large software systems. Less attention has been paid to the problem of visualising other types of systems, in particular, the systems-of-systems discussed in section 2.2. Examples of systems-of-systems are Defence C2* systems, the Internet and the large-scale intranets used by many organisations. These systems are built up from many different hardware and software elements, distributed over wide geographical areas that rely on various mechanisms for communication between elements. The large, complex nature of these systems and their lack of a physical context in which to visualise them, means that they suffer from similar visualisation problems to software systems. The research work described in this report investigates the possibility of tailoring the visualisation techniques developed for the software domain for the visualisation of systems-of-systems.

3.4 Exploratory Nature of Computer-Based Visualisation

One of the important benefits of computer-based visualisation is that descriptions can be made to be highly flexible, allowing the user to adapt the description based on their

particular needs. This then allows the user to interact with the description. The description provides information about the system the user wishes to understand, and, based on the information conveyed to the user, the user may then alter the description to focus on particular features of the system. Used in this way, computer-based visualisation is a useful approach for exploration of the system being visualised. However, its effectiveness relies on the user's understanding of the visualisation tool being used, what the tool can do, and how the tool's functions best help the user to visualise the system in the context of the task the user is performing. For a generic visualisation tool that may be used by different users wishing to view the system from different perspectives, endowing the tool with the capability to be easily customised to user needs and the domain context is important to save time and reduce the overhead needed to use the system. For instance, some users may regularly require a number of selected system views that highlight information of specific interest to their domain tasks. These views may be different for each user, depending on their preferences and tasks the views support. In general, current visualisation tools do not provide support for their easy customisation and configuration based on the domain context and user and task needs.

3.5 Problems and Limitations

In addition to the issue of customising tools to user needs and a domain context, there are other limitations to many of the current visualisation approaches. Some of these limitations relate to the problems summarised in Table 2.1 above. Extending the idea of the need for customisation of views, tools should provide support mechanisms for guidance and ease of use. Multiple users using the same visualisation system for different tasks will require guidance in system use to obtain the most suitable descriptions of the system. This need for the visualisation tool to cope with multiple users is linked to the problem of system complexity. Large systems will have different levels at which the system may be viewed, and users may need to view the system from different perspectives. For instance, a person at the technical level may need a lower-level, more-detailed view of the system than a person at the management or command level.

The importance of noting the visualisation system's intended goals and context of use has been acknowledged by Price, Baecker et al. (1993), in discussing the visualisation of software. They also suggest that the issue of scope is the "largest impediment to the use of software visualisation" and that "many issues of scalability remain to be solved". This is of particular import in relation to this report, as visualisation is intended to be an aid to understanding large complex systems. Other issues highlighted by Price et. al. are the form of system visualisations (whether they be graphical or auditory), the granularity of visualisations, the methods for specifying visualisations, facilities for navigation and interaction with visualisations and the need for empirical evaluation of visualisation techniques.

Duffett and Vernik (1997) report on a number of important issues relating to the usefulness of the *Netmap* visualisation tool. For instance, they report that there is a

problem with scalability of the visualisations, with large information sources being much more difficult to visualise using one visualisation. In addition, the ability to zoom-in for more detailed views of information can result in a loss of context. Grouping algorithms, that grouped system elements according to certain rules, were found to be useful, however they were inflexible and could not be adapted by the user. Other limitations were a small number of display views, no tools for displaying statistical information, no usage guidelines, no recording tools for auditing or automation, no scripting facilities, and no facilities for attention direction. One of the primary limitations was the large amount of overhead required to set up the tool so that it could be used for the particular application domain (software visualisation). This overhead includes defining the visualisation requirements, writing scripts for accessing, filtering and integrating the underlying information being visualised by Netmap, and tailoring the resulting visualisation to the specific need of the user.

Computer-based visualisation has many potential benefits, however, as discussed here, there are a range of issues that need to be addressed if it is to provide an effective form of description. These issues include the requirement to consider user needs within an overall domain context, information assimilation/integration, need for customisation and adaptation, set up costs, and the need for an underlying model of the system being described so as to support information integration and the generation of consistent multi-perspective views. Chapter 4 will discuss an approach to visualisation and description that encompasses these issues.

4. Integrated Visualisation and Description Approach

In this chapter the Integrated Visualisation and Description Approach is discussed. The approach revolves around the use of a *Composite Systems Model* which provides access to integrated system information and is a foundation for the creation of customisable and adaptable computer-based descriptions of complex systems. Based on a conceptual model of the description process initially proposed by Vernik (1996), the approach takes advantage of new techniques in component-based software engineering as well as software agent technology.

4.1 The Concept of a Description Process

To facilitate the description of a system, we introduce the concept of a description process (Figure 4-1). The description process acts as a filter between the information space and the user and helps to create focused descriptions of the systems. This is particularly important for viewing high-level system descriptions that might require the integration of information from several information items. The descriptive information created by the description process must be customised to the particular needs of the user. Therefore the description process includes the tasks of discovering the information needs of the user, selecting the information items required to fulfil these information needs, selection of an appropriate format for the description, and adaptation of the description to the particular needs of the user.

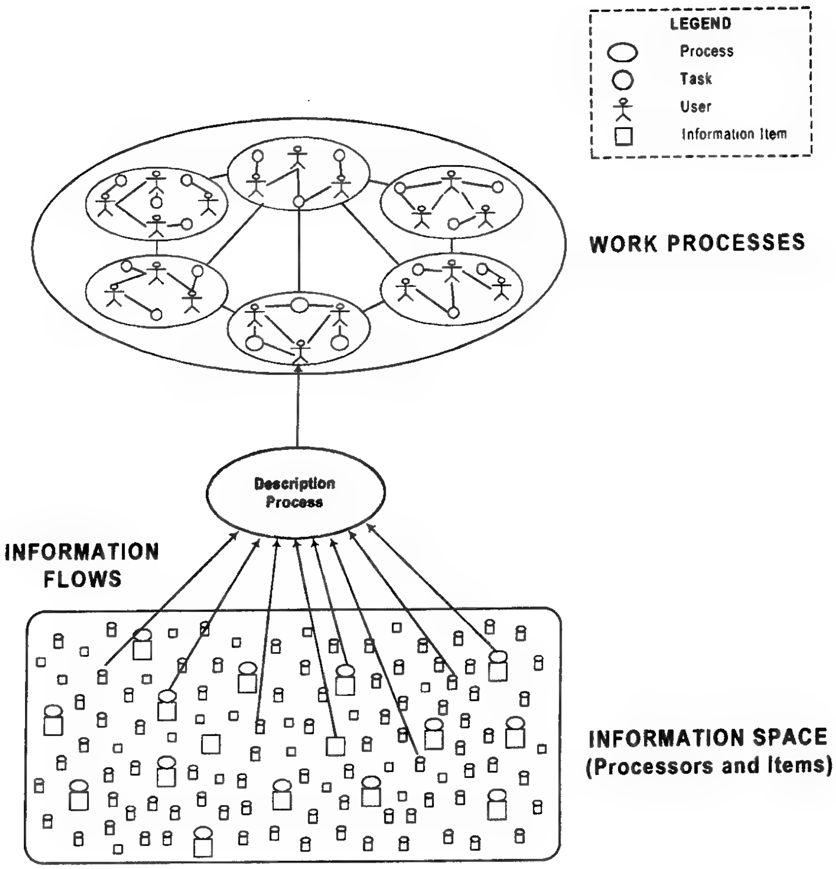

Figure 4-1 - Concept of a Description Process

4.2 Conceptual Model of Description Processes and Products

Figure 4-2 shows the four key subprocesses that constitute the description process (*facilitation, provision, mediation* and *use*) and the system descriptions that are created as a result. It should be noted that there may be several descriptions created by the description process. Each description may provide support for the other descriptions or provide alternative views of the system at different levels of abstraction.

The provision process accesses the appropriate information items from the information space, extracts the specific information required from these items and integrates it into a system description that is presented to the user. This system description may be shown as a family of views of the information space, rather than as a single view, as it may be impossible to show all the required information within one view. These views may show different aspects of the system or show the system at different levels of abstraction.

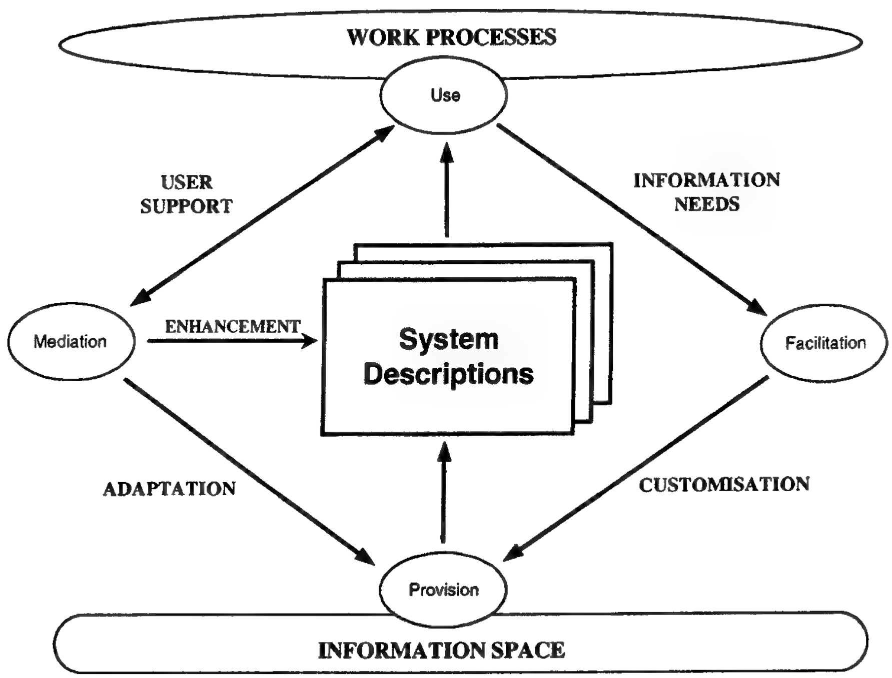

Figure 4-2– Refined concept of a description process

The use process includes the human information processing and cognitive processes such as perception and problem-solving. It involves the user establishing the goal they wish to fulfil, the information required, and the form the information is to take (the description). It also involves the interpretation of the descriptions and evaluation of the descriptions in relation to the original goal.

Many current visualisation tools and approaches support only the provision of information with the focus being the production of new kinds of representations of information. However, the representation is just another source of information. What is important is who needs the information, what they need it for and how they interact with it. These issues are considered through the inclusion of facilitation and mediation processes.

The facilitation process involves the elicitation of the user's information needs, either through the explicit statement of the need by the user, or through some interaction between a human or computer-based facilitator. This facilitation process uses the knowledge of the information needs to select the appropriate information items from which it can obtain the information required to meet the user's needs and to specify a custom description for viewing this information.

The mediation process gives the user the ability to adapt the system description to their specific need. In most cases the facilitation process will give an approximate solution to

the user's information needs. It will be necessary for further adaptation of the description through the provision of a flexible user interface that allows the user to adapt descriptions, through direct manipulation or some other means of interaction. The mediation process also includes explanation of the information displayed by the system description, advice on how descriptions may be adapted to specific purposes, and can provide enhancements to descriptions through attention direction (e.g. annotating relevant features on the description).

The mediation process supports the exploratory nature of computer-based visualisation, allowing the user to experiment with various forms of system descriptions in the search for a description best suited to the user's information needs. However, finding an appropriate description can be a hit-and-miss affair and a user can potentially waste much valuable time experimenting with a number of different views of information. The facilitation process saves time by providing the user with custom descriptions of information based on the user's requirements. The facilitation and mediation processes complement each other - the facilitation process provides automatic custom descriptions that can be adapted through the mediation process to suit the user's specific information needs.

4.3 Integrated Visual Descriptions

Section 3.1 highlighted the need for flexible methods of interaction between users and computers for the customisation and adaptation of computer-based visualisations used for system description. Integrated Visual Descriptions (IVDs) (Vernik 1996) are classes of information items that use an integrated set of computer-based visualisations to describe systems of interest. They are generated from an underlying model of the system, a Composite Systems Model (discussed in Section 4.4), and are designed to support the consistent representation and presentation of descriptive information. Moreover, they provide a flexible means of customising and adapting information to suit individual needs. This section discusses the characteristics and main features of IVDs.

4.3.1 General Characteristics

The IVD concept is based on the use an integrated set of topographic and supporting views (see Figure 4-3). The topographic views are system overviews based on compact, spatial representations of sets of system elements. The arrangement of these elements is based on a set of rules which can be defined in terms of the structural relationships captured in the Composite Systems Model (see Section 4.4). Topographic views provide central visualisations that can be readily adapted to describe those characteristics of the system that are of importance to the user. The compact representation allows other information to be integrated and viewed in context. For example, lines may be overlaid onto these 'base representations' to describe important information flows between system elements. The shading or colouring of the elements may describe the values of particular attributes of interest (e.g. those entities that have most recently changed, or those that have complex internal structure). The topographic

views provide a basis for integration of visual representations and are designed to provide flexible and scalable representations that can be readily customised and adapted to user needs. Supporting views provide alternative representations of the information provided in the topographic views. They also allow for additional information to be provided. For example, the user may find it more understandable to view particular relationships of a subset of the system elements shown in the topographic view as a tree or cluster graph. A chart may provide an effective means of describing the trends in a set of numeric attributes for particular elements. A text view might provide a link to some underlying textual representation of an element (e.g. a section of the source code or documentation about the element).

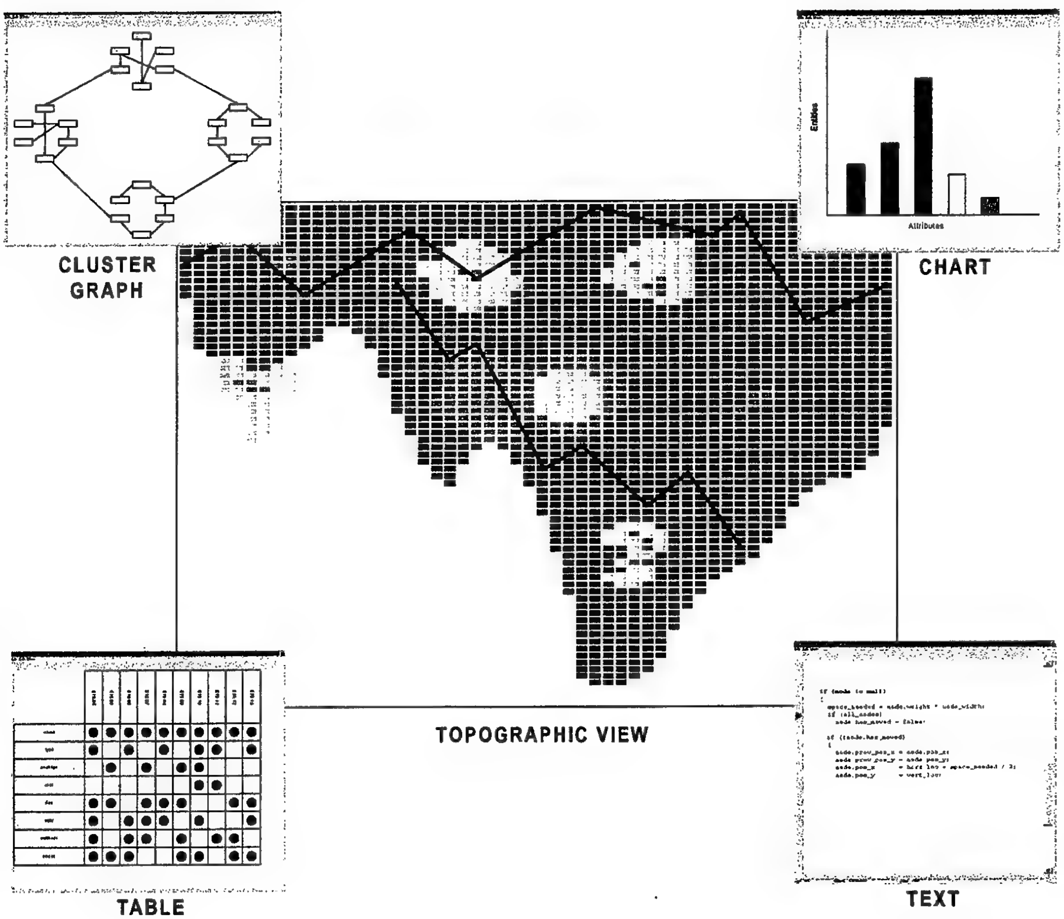

Figure 4-3 – Concept of Topographic and Integrated Supporting Views

4.3.2 Topographic Views

Topographic views are based on a geographic map metaphor with the topographic view providing a spatial representation of the system of interest, with the shape of the entire layout and colours and shapes of elements providing information about important features of the system.

Figure 4-3 shows an illustration of a topographic view and its supporting views. A topographic view (centre) is the primary view of the system and sets the context for the supporting views (shown surrounding the topographic view). Several topographic views may be required to support different user information needs and viewpoints required during the performance of a task and each of these may have a number of supporting views. The rectangles in the topographic view illustrated may represent a particular type of element in the system that is of interest to the user. For example, they may be software components in a software system, nodes in a computer network, or people in an organisation. Elements need not be represented by rectangles only, but may be represented by different shapes, each shape reflecting some property of the element, such as its type. Each representative shape may also be coloured according to some property of the element. For instance, the topographic view above might have elements coloured yellow to indicate a certain level of complexity of the software component. Relationships between elements may also be overlaid on the topographic view. For instance, the zigzag lines shown in Figure 4-3 might indicate control paths showing the flow of execution through a software system. Examples of different topographic views can found later in the report (see Figure 5-1 - Figure 5-3).

Figure 4-4 shows how a topographic view might be specified. A composite topographic layout can be specified by combining various layout arrangements to represent particular system characteristics. For example, at each level of the system view, classes of system elements of interest are identified and laid out according to a set of layout rules based on particular attributes of system elements and relationships between them. There are several options available for laying out the elements at each level – linearly (either horizontally or vertically), as nodes in a graph, or in cluster, grid or outline layouts. Users may customise other layout features such as the space between elements and the element alignment. The IV&D Approach allows the number of layout methods to be extensible to take advantage of any new layout methods that are devised in the future (i.e. by "plugging in" new layout components). The large number of options available to users for selection of elements to be viewed using different layout methods at several system levels results in a multitude of different layout combinations and topographic views being available, giving users and analysts a high degree of flexibility for developing system descriptions.[1]

[1] See Figure 5-2, 5-3 and 5-4 for examples of how these generic concepts can be used to produce dramatically different topographic views.

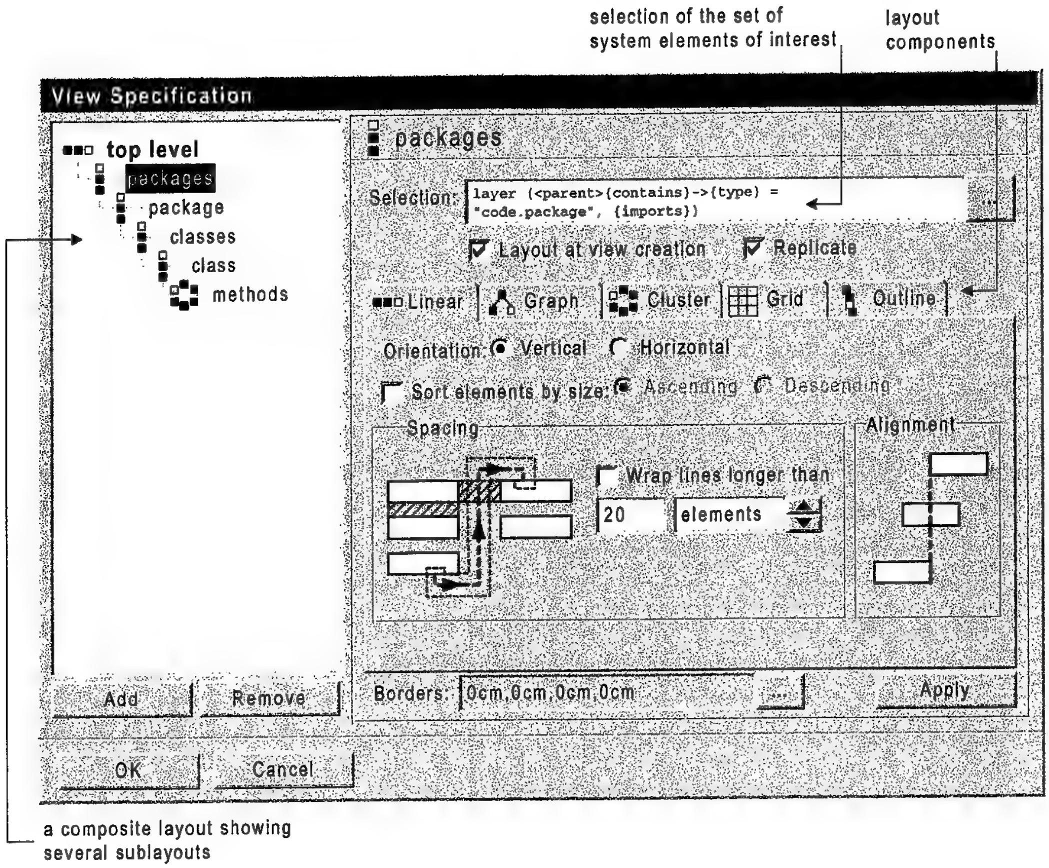

Figure 4-4 - Specification of Topographic Views

4.3.3 Supporting Views

Supporting views are used to support the topographic view by providing additional information about the system, system elements and their relationships. Often this additional information will provide more details on areas highlighted in the topographic view and is presented using more traditional visual representations such as chart views, table views, graph views and text views. For example, chart views can be used to show trends in attribute values of elements. Tabular views can be used to compare this information in numeric form and for generation of higher-level measures. Graph views allow the user to display information about the relationships between sets of elements to highlight key features (e.g. element clusters). Text views provide a link to textual descriptions of elements (e.g. software source code).

The topographic and supporting views complement each other - topographic views are good for getting an overview of the system (or subsystem) that is of interest and highlighting interesting features of the system based on selected attributes and relationships of the elements of the system. The topographic view also provides the

context for the supporting views that can be used to look at these features in more detail. Together the two types of views provide a means for visualising the features of the entire system. Consistency is maintained between views by basing all views on information contained in a model of the system being described. This model is called the Composite Systems Model. This model is discussed in Section 4.4.

4.3.4 Example of IVD Use

To illustrate how the topographic and supporting views might be used consider Figure 4-3 which is an example IVD of a system, in this case we will suppose it is a software system. A spatial description of the system has been created by laying out the rectangles (representing software components, say) according to predetermined layout rules. The layout in this example might be a linear layout from left to right where elements are arranged based on their 'use' dependency. For example, an element in the third column could use another element in the same column or to its right. The shape of a topographic view is significant in that long tails might indicate high levels of component coupling. The shape also reflects the design approach used, levels of reuse etc. Feature overlays are used to highlight different regions of interest. The yellow region might indicate areas of high complexity, red might indicate defects in the elements, blue might indicate the descriptiveness of the code. The lines flowing between elements might highlight control flows through the system as the different components call succeeding components in the chain.

The relation overlays (in this example the control flows) may be animated as might other feature overlays. One may, for example, wish to show flows of execution through a software system during a particular test sequence. In general, animation will show how features and relations between features may change over time.

Supporting views enable the user to look at the features of the system in more detail. For instance, the cluster graph view shows a view of the dependencies between elements in one region of the topographic view. The chart view allows the user to compare the size of the different regions in the topographic view based on the number of elements within each region. The table view allows a numeric comparison of element attributes and the text view shows the specific lines of source code associated with a particular module.

Although this example has focused on software systems, the concepts apply equally to other types of systems. For example, the IVD of Figure 4-3 might represent a large intranet system where the elements represent the nodes and where they are arranged based on the relative connections to one another. The yellow colour overlays might represent areas of most activity during a particular period of time and the red overlay might represent node failures. The lines might represent flow of information.

It is important to note that the consistency between views is maintained because all views are based on a single model of the system, the Composite Systems Model. The Composite Systems Model is a reference model that integrates all system information that is relevant to the needs of a set of users. This is discussed in the next section.

4.4 The Composite Systems Model

In many cases, a system description will need to be made up of a number of different views to gain an adequate description of the system to suit the users' information needs. It is important that each of these views is consistent with the other views of the system. In the Integrated Visualisation and Description Approach all views are generated from a Composite Systems Model that provides a basis for integration of all information used to describe the system.

The Composite Systems Model (CSM) is a named set of system elements, each element having a set of attributes and relationships with other elements. As discussed in Section 2.1 a system, **S**, may be described mathematically as an ordered pair of sets, (**E**, $\mathbf{R}_\mathrm{E}$), where E is a set of elements making up the system, and $\mathbf{R}_\mathrm{E}$ is the set of relationships between the elements of set **E**. This can be described by the equation

$$\mathbf{S} = (\mathbf{E},\mathbf{R}_\mathrm{E}). \qquad \mathbf{E} = \{\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3,\ldots \mathbf{e}_n\},\ \mathbf{R}_\mathrm{E} = \{\mathbf{r}_{\mathrm{E}1},\mathbf{r}_{\mathrm{E}2}, \ldots \mathbf{r}_{\mathrm{E}m}\}$$

A model of a system is a simplification of the system, where only certain selected features (elements and relationships) of the system are of interest. Therefore a model of a system **S** may be denoted by

$$\mathrm{M}(\mathbf{S}) = (\mathbf{E}',\mathbf{R}_{\mathrm{E}'}) \text{ where } \mathbf{E}' \text{ is some subset of } \mathbf{E}.$$

A model where the system elements and their relationships are of interest is called a *structural model*. For clarity, the structural model of a system, **X**, consisting of a set of elements, E(**X**), is denoted by

$$\mathrm{M}_\mathrm{S}(\mathbf{X}) = (\mathrm{E}(\mathbf{X}),\mathbf{R}_{\mathrm{E}\,(\mathrm{X})}).$$

One can also consider a model of a system where the set **E** is the set of attributes that describes a single element, **X**. We might then describe the system in terms of this *atomic model*, denoted by

$$\mathrm{M}_\mathrm{A}(\mathbf{X}) = (\mathrm{Attr}(\mathbf{X}),\ \mathbf{R}_{\mathrm{Attr}(\mathrm{X})}),$$

where Attr(**X**) is the set of attributes that describes the element, **X**, and $\mathbf{R}_{\mathrm{Attr}(\mathrm{X})}$ is the set of relationships between these attributes.

There is often a need to combine structural and atomic models. These *composite models* allow system attributes to be described within the context of relationships that define the structure of the system. Combining the latter two equations above provides a formal definition of a composite model

$$\mathrm{M}_\mathrm{C}(\mathbf{X}) = ((\mathrm{Attr}(\mathbf{X}),\ \mathbf{R}_{\mathrm{Attr}(\mathrm{X})}),\ \mathbf{R}_{\mathrm{E}\,(\mathrm{X})}).$$

Composite models provide a basis for developing higher-level descriptions by combining and processing information pertaining to the attributes of other (often

subordinate) elements. In this way, features of the entire system can be described through combinations of attributes and relationships of lower-level system elements.

The IV&D Approach uses a Composite Systems Model to model the referent systems in terms of only those features that are of interest to a specific set of users. It provides the basis for integration of system information from a large number of disparate information items into a consistent model. All descriptions generated by the IV&D Approach are based on this Composite Systems Model, ensuring consistency between descriptions no matter what method of description is used.

4.5 Implementation

The IV&D Approach is realised through the application of a *component-based software engineering* (CBSE) methodology to develop a collection of generic component frameworks and domain-specific software components that are used as building-blocks to build Integrated Visualisation Environments (IVEs). An IVE is a domain-specific software environment that is used to generate Integrated Visual Descriptions (IVDs) of a system within a specific user domain. The IVDs produced are based on families of topographic and supporting views. Information conveyed by IVDs is obtained from a Composite Systems Model, managed by the IVE, which integrates information from a disparate set of information items that describe aspects of the system of interest. An IVE is assembled from components that are selected according to the requirements of users and the user domain. Component frameworks provide the necessary infrastructure for the interoperation of the components (see Figure 4-5). In this way, the CBSE methodology allows for customisation of the IVE for specific domains or sets of users.

Deployment of IVEs into their domain environment is supported through the use of software agents. These agents perform various functions such as mapping the information space, automating the evolution of the Composite Systems Model, and supporting users through customisation and adaptation of system descriptions. The following sections describe the architecture of an IVE and the supporting agents in more detail.

4.5.1 Integrated Visualisation Environments

Integrated Visualisation Environments (IVEs) are the central elements of the IV&D Approach. An IVE is specified as a set of component assets, comprised of both components and component frameworks, and is developed using a component-based software engineering approach. This approach enables components to be added, removed and replaced in a "plug-and-play" manner so that the IVE can be customised to the needs of its users. New view types, modelling tools, agent components and automation features can all be integrated into a customised IVE so that it may support different users effectively.

A component framework is defined as "a software entity that supports components conforming to certain standards and allows instances of these components to be

"plugged" into the component framework" (Szyperski 1997). A framework therefore defines a common environment for components, specifying methods of interfacing between components and providing some level of functionality that components both inside and outside the framework can build on. It is important to note that a framework may itself be a component of a system where it is subject to the same plug-and-play assembly technique employed with other components.

Figure 4-5 shows the conceptual architecture of an IVE with the major frameworks and components that make up the IVE. Black dots and lines indicate plug-in points and connections between components and the components frameworks that provide the infrastructure for component interoperability. White dots indicate plug-in points where new components may be connected into the frameworks. Descriptions of each of these components and frameworks is provided in the following sections.

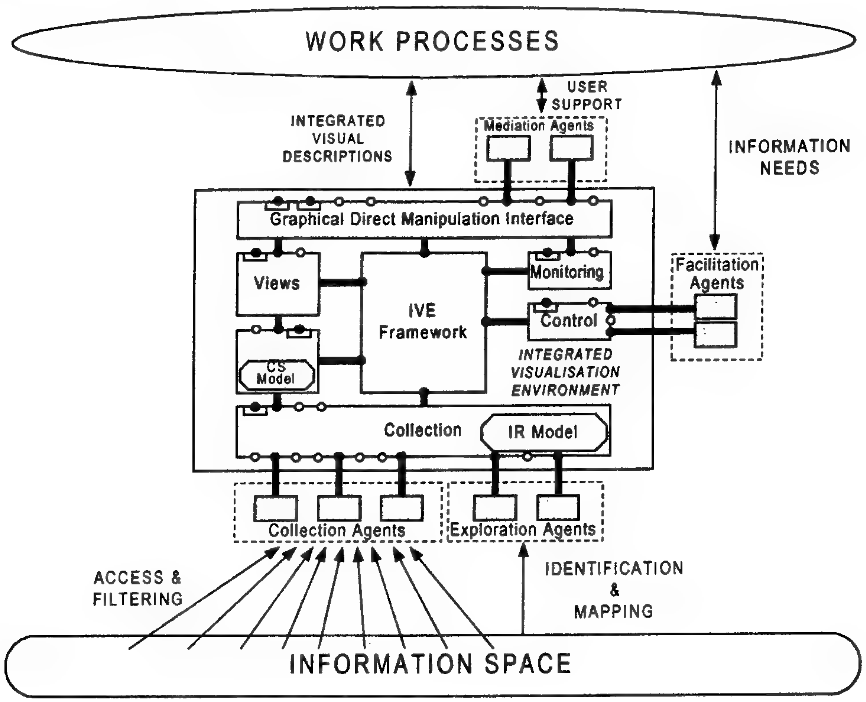

Figure 4-5 – The IVE Conceptual Architecture

4.5.1.1 *The IVE Framework*

The IVE Framework is the central infrastructure into which all other component assets connect. It provides standard component interfaces and services that are common across all components, such as persistent storage, component registration, etc.

4.5.1.2 *The Views Framework*

The Views Framework provides the infrastructure for all view components within an IVE. This includes support for view layout, specification, window management, view

integration etc. Examples of standard view components in this framework are the Topographic View, Table View and Chart View. The framework also allows new view components to be "plugged in" and fully integrated with existing views. The Views Framework connects to the Modelling Framework as well as the IVE Framework to allow each view full access to the CSM.

4.5.1.3 The Collection Framework

The Collection Framework provides the infrastructure for "pluggable" Exploration and Collection Agents as well as managing the Information Resource Model (IRM). The IRM acts as a map of the information space, and is used primarily by Collection Agents to access and filter information of interest from information items when building a CSM.

4.5.1.4 The System Modelling Framework

The System Modelling Framework is a generic component that maintains system models. In an IVE environment, this framework maintains both the Composite Systems Model and provides support for modelling the information space through the IRM. The System Modelling Framework is primarily concerned with storing and managing system elements, element attributes and relations between elements. It also provides a high-level query language for accessing system models.

4.5.1.5 The Direct Manipulation Graphical Interface

The graphical user interface framework supports all visual components of the IVE in presenting a consistent and rich interface to the user. This framework is based on modern graphical user interface principles with an emphasis on direct manipulation of on-screen elements via drag-and-drop gestures. In collaboration with the Control Framework, it also supports full undo/redo of user commands.

4.5.1.6 Monitoring and Control Frameworks

The Control Framework provides the infrastructure to support the automation of various IVE processes. For example, it provides facilities such as a scripting language that can be used to automatically generate customised views or support the generation and evolution of the Composite Systems Model.

The Monitoring Framework enables user actions to be logged in relation to specific tasks. This type of monitoring is an invaluable part of effective research into how the tool is being used and where it might be optimised for particular tasks (Phillips and Vernik 1997). The usage data collected by this framework can directly support tool optimisation by taking the commands executed by the user during a task and generating an equivalent script. The monitoring functions provided by this framework are also available to Mediation and Facilitation Agents that need to track what the user is doing in order to automatically tailor the tool to that user's needs.

4.5.2 Agent-based IVE Support

Software agent technology has become a popular topic of research in recent times. This popularity has lead to a large number of commercial software agents being made available, particularly for use as information filters on the Internet and WWW. Unfortunately the popularity of software agents has resulted in the word "agent" becoming a "buzz-word", and it has become fashionable to call any piece of software a software agent. The definition of the term "software agent" has as a result become blurred. Rather than provide a discussion of all the possible definitions and characteristics of agency, the reader is referred to Bradshaw (1997), Jennings and Wooldridge (1996) or Nwana (1996) for more information.

For the purposes of this discussion, IV&D agents are autonomous software components that support IVE system elements[2] in the execution of tasks. By autonomous it is meant that the agent does not require continuous interaction with its client elements in order to support their tasks. Autonomy is considered to be a minimum condition for agency. Other characteristics of agency may be required by the software agents in order to support the IVE and its users, such as intelligence and mobility. However the existence of these characteristics are considered more important as a measure of the capability of an agent to achieve its primary goal of supporting clients' tasks, rather than a measure of its agency.

Agents are an important part of the IVE infrastructure as they aid in deploying an IVE into its environment. The component-based software engineering approach is used to create a component system architecture that provides generic infrastructure required for the assembly of IVEs. This approach also considers how an IVE will be deployed. The approach requires domain knowledge for both assembling an IVE and for selecting and programming the agent-based components. Some knowledge about the domain is embedded in domain specific components that may be "plugged" into the IVE framework, for instance specific views components may be required to support user domain tasks. However, low-level domain knowledge, such as the physical location of specific information resources, the location of information items within those resources, user preferences and task-specific information must be gathered in order to successfully deploy an IVE into the user domain. Agent components will be used to assist the deployment of IVEs into user domains by capturing this low-level domain knowledge.

The agents that support the IVE are categorised into types that are roughly associated with the description subprocesses discussed in Section 4.2. These types are Exploration and Collection Agents (associated with provision), Facilitation and Mediation Agents.

4.5.2.1 Exploration Agents

Exploration Agents are the first of two types of agent associated with the provision process. They are probably the most generic of the agents supporting the IVE. Their

[2] An IVE system element could be a user or an IVE component.

role is to seek computer-based information items within the information space, mapping out the location of the items and recording the types of the items. This information is recorded in the Information Resource Model (IRM) where it can be accessed by Collection Agents. Exploration Agents, like the Collection Agents described below would be categorised as information agents in the terminology of Nwana (1996).

4.5.2.2 Collection Agents

Collection Agents extract descriptive information from information items in the information space. This descriptive information is then used to update the Composite Systems Model. Collection Agents identify the location and type attributes of information items using the IRM. These details are used by Collection Agents to determine which information items they can extract information from and the location of these information items.

4.5.2.3 Facilitation Agents

Facilitation Agents provide users with support in producing custom system descriptions. They are able to do this by having knowledge of the domain, user tasks, and users embedded in their knowledge base, or by learning about the domain, tasks and users through direct interaction with users. This knowledge is then used to determine the specific information that must be conveyed by a system description in order to support a specific set of user tasks. Facilitation Agents also have knowledge about what visual representations may best be used to convey this information. They can then automatically generate description specifications or, given knowledge of previous user preferences and use of the IVE, select previously generated specifications that match the users' current information requirements.

Faciliation Agents also supports the evolution of the CSM by determining users' information needs in terms of system elements, attributes and relationships that must be described to the user, and arranging for this information to be collected and integrated into the CSM via the provision process.

4.5.2.4 Mediation Agents

Mediation Agents support the IVE in two ways. Firstly, Mediation Agents support the user by providing explanations of features of the IVE similar to a wizard providing help on the use of a software application. Mediation Agents also support the user through explanation of the descriptions created, identifying noteworthy features of the descriptions, and explaining their possible meanings. Attention direction has been noted by a number of authors as being a desirable feature of visualisation tools (Rogers 1995; Duffett and Vernik 1997). This may be extended to include enhancement of the system descriptions by highlighting features within the descriptions using colours or methods such as circling the features of interest. This requires the Mediation Agents to have knowledge about user preferences, the context in which the descriptions will be used and the domain of use. Mediation Agents also require some knowledge of descriptive methods available, have the ability to recognise features of the descriptions and are able to support the user in the interpretation of these features.

The next section looks at the IV&D Approach in more detail by providing examples of visualisation scenarios

5. Applying the IV&D Approach

This section provides examples of how the IV&D Approach might be applied in several different application domains. This will illustrate the use of the approach in more detail as well as demonstrating the scope of the technique. The IV&D Approach is a generic approach to systems visualisation and can be applied to produce a variety of visual representations to support the needs of users in various application domains. The applications described here are the visualisation of heterogenous software systems, C2* systems-of-systems visualisation, and visualisation of enterprise architectures.

5.1 JCSE Software Scenario

This first example shows how a simple topographic layout can provide for effective software systems descriptions. The example focuses on the visualisation of software for one of the key Defence Command Support Systems – the Joint Command Support Environment (JCSE). The JCSE software is a heterogeneous COTS-based system comprising of a variety of different types of source code. The information space for the JCSE software consists of approximately 90 classes of source code files. The instances of each of these classes can number in the thousands. This large number of different classes and instances make it very difficult to get an overview of what is there in order to get an understanding of the key characteristics of the system. Appropriate visibility of the system is important for developers, managers, acquirers, capability planners, and IO analysts. The types of questions they might want answered are

- What development technologies have been used?
- Where has most work been done since the last release?
- Are we converging to a set of key technologies?
- Which parts of the system are most stable?
- How big are the different product artifacts?
- What are the relationships between the artifacts?

Let's consider a scenario where the IV&D Approach is used to support the JCSE configuration manager. The configuration manager might initially need an overall picture of the system. A facilitation agent might be invoked to automatically establish a CSM for the system based on user input and a knowledge base of configuration management tasks. Exploration Agents then map the information space and Collection Agents capture information about source files in terms of a set of attributes that

describe particular characteristics such as the type of file, size, date of the last modification etc. The information captured is then integrated into the CSM. The configuration manager might then use pre-defined integrated visual descriptions to gain visibility of the system or define/modify specifications for his/her own customised views.

Figure 5-1 shows an example of a set of views which might be used to give an overview of the JCSE software system elements.[3] Here the elements we are looking at are source code files. These files have been laid out in a linear layout, grouped according to file type, and ordered from left to right by the number of files of each type. Different layouts may be used depending on user preference or on the information that must be conveyed by the description. Different layout specifications will result in the topographic view exhibiting different shape and pattern features. Such features may convey important local and/or global properties of the system.

Once the desired layout has been created, various attributes of the system elements can be highlighted by using the Features Overlay Panel (seen on the left of Figure 5-1). This panel shows the attributes and relationships of the system elements that can be overlaid on the topographic view. In this example the system elements shown are all source code files. File attributes information such as *type, lastModified, bytes, lines*, and *path* may be overlaid on the topographic view. In this instance, the type of the system elements has been selected as being of interest to the configuration manager. Colour overlays have been used to add type attribute information to the topographic view. Colour mappings for each source code type may be selected by using the Mapping Table in the Features Overlay Panel. In the example, overlaying attribute information using colour enables the user to easily identify the types of files shown in the topographic view and make a comparison between the numbers of files of each type in the JCSE software system. For instance, Figure 5-1 shows that the JCSE contains only two Fortran files, shown on the left of the view panel in olive green. On the right of the view panel we can see that there is a large number of Ada files, which represent primarily the older legacy part of the system.

---

[3] Although this IVD was based on real data obtained from the JCSE software, Figure 5-1 is not an exact representation of the actual configuration at this time.

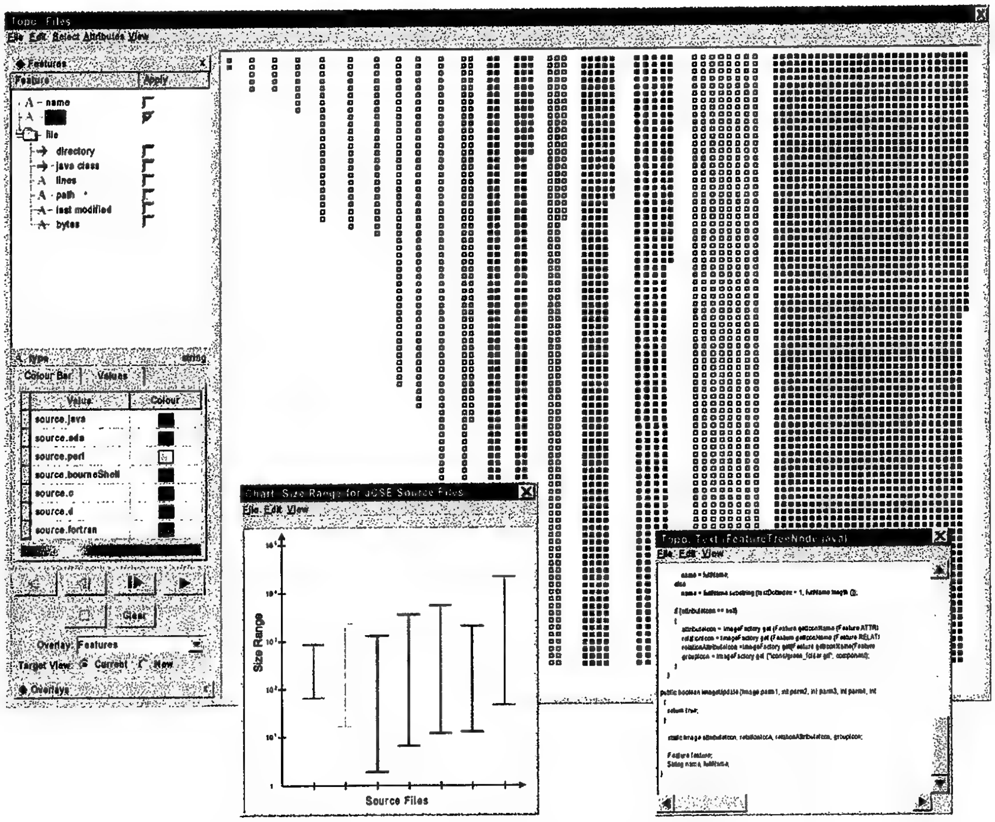

Figure 5-1 – An Integrated Visual Description showing topographic and supporting views of JCSE Software

The configuration manager may use supporting views to examine the attributes of the code files in more detail. Figure 5-1 shows examples of two supporting views. The supporting view on the right is a text view that allows the user to view the actual text of particular code files. The view on the left is a chart view showing the range of the sizes of source code files for the seven source code file types highlighted in the topographic view. The bars have been arranged according to their order in the topographic view and have also been coloured to correspond with the colours used in the topographic view. A logarithmic scale has been used for this chart due to the large disparity in size between the code files. This view allows the configuration manager to assess whether the size range of each of the file types falls within their respective norms. For example, macro files would be expected to be small. More conventional programming languages, such as C and Ada, should not have lots of small files (which might indicate poor design abstraction). Similarly, conventional programming languages should not have files that are excessively large. Another useful view for the user may show the size distributions for each source code type.

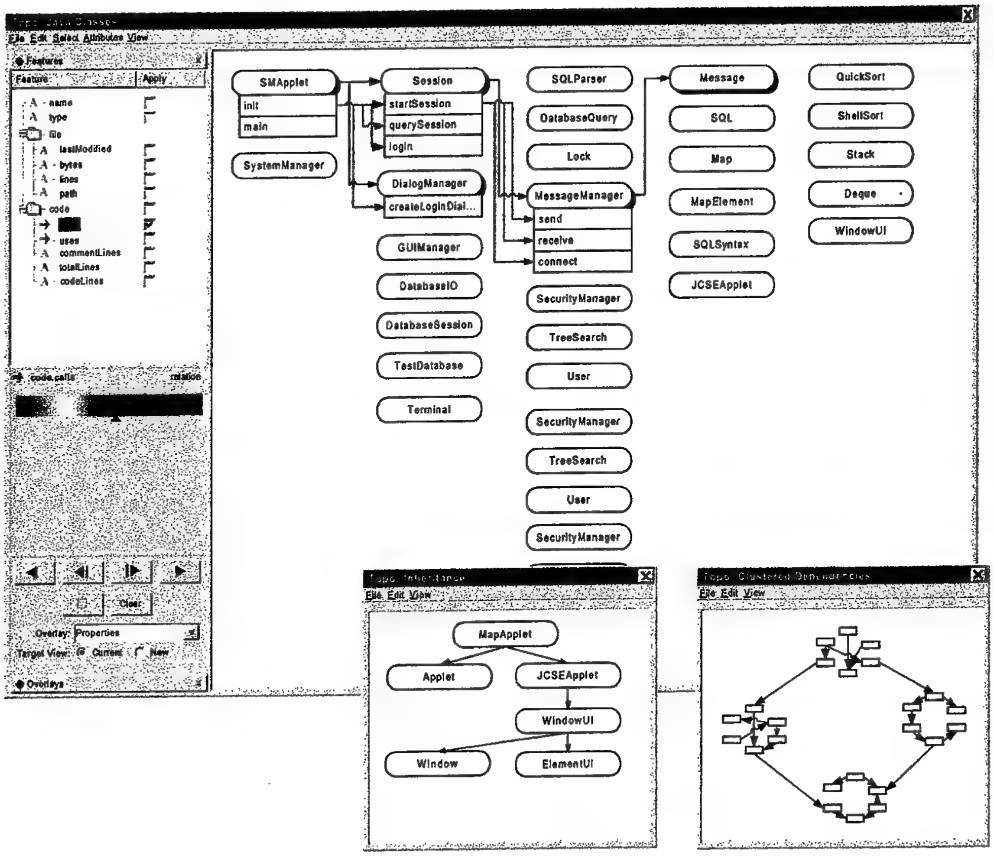

Figure 5-2 – Visualisation of Java Software

Topographic views may be used to view systems at different levels of abstraction and detail. For example, analysis may show that Java software has been most recently added to the JCSE software. This may warrant closer examination by the configuration manager. In order to gain an understanding of the semantics of the Java software, information needs to be extracted from the software to identify the code elements and their relationships (e.g. Java packages, classes, methods, etc.). A new Collection Agent, a Java Parse Agent, could be assigned the job of extracting semantic information from the Java source files in the information space and updating the CSM[4] with this information. The configuration manager may then invoke a new topographic view (Figure 5-2) to show this more detailed view of the Java software. This view is arranged to show the Java classes laid out according to the uses relationship so that classes only

---

[4] It is important to note that the CSM need not represent every element and attribute of the system. It need only model the system elements and attributes of interest to the user for the task at hand. The CSM will generally evolve over time, with new system elements and attribute information being added to the model as needed through the use of Exploration, Collection and Facilitation Agents.

use other classes in the same column or those on the right (the specification for this topographic layout is shown in Figure 4-4). Using this layout rule ensures that the topographic view is a compact spatial representation of the Java software. A feature overlay has been used to show the *calls* relationships between methods associated with one Java class (SMApplet) and methods of other Java classes. Methods are only shown on demand when the user wishes to see the calls relationships between methods. This is one of the scalability features of this view. Further supporting views have been invoked to show details such as the inheritance relationships between the Java classes and clustered dependencies.

This scenario has illustrated the key features of the IV&D Approach for the configuration management task. In particular it illustrates the use of topographic and supporting views for the description of a real heterogeneous software system. The following visualisation scenarios show the use of the same generic approach for the visualisation of systems in two very different systems domains.

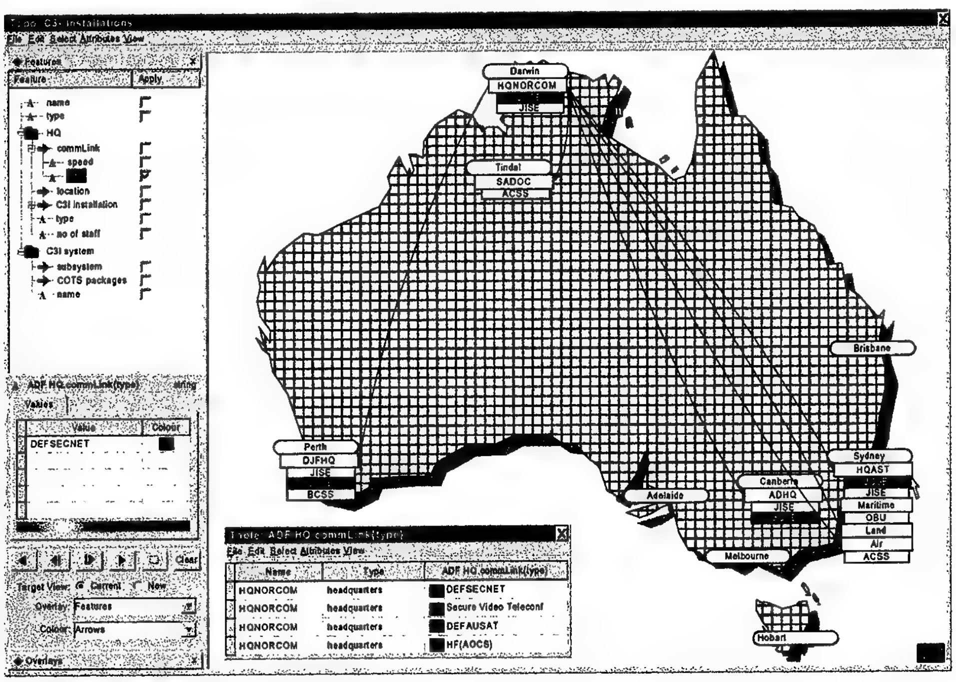

Figure 5-3 - Visualisation of C2 Systems-of-Systems*

5.2 Visualisation of C2* Systems-of-Systems

As was discussed in section 2, there is a need for visualisation techniques to deal with the problem of visualising increasingly complex systems. A new class of system that is emerging is the so-called system-of-systems (SOS). One example of a SOS is the Australian Defence Force C2* capability which comprises various Command Support Systems (CSS) located at different headquarters around Australia. Examples of these

CSS systems are the Joint Command Support Environment (JCSE), Battlespace Command Support System (BCSS) and Air Command Support System (ACSS). In terms of the overall C2* capability, these systems are, in turn, supported by other systems such as the various Defence communications systems.

A description of this C2*system-of-systems is shown in Figure 5-3. It highlights the novel way in which information may be laid out in topographic views. A geographic grid layout has been used to position system elements according to their geographic location. A vertical layout has been used at each of these locations to show the cities, the major headquarters located within these cities, and their associated Command Support Systems. This layout may be specified using the dialog shown in Figure 4-4. A map of Australia has been overlaid to set the context for users.

Two feature overlays have been used in this IVD. Red has been used to highlight the JCSE installations and a relationships overlay has been used to highlight the DEFSECNET communication links between HQNORCOM and the other headquarters. Various other information of interest can be overlaid in this way. For example, one can overlay information showing the number of staff stationed at each headquarters, or the COTS packages being used at each headquarters.

5.3 Visualisation of HQAST Information Flows

In 1996/97 Aspect Computing was commissioned to undertake a study of the information flows in Headquarters Australian Theatre (HQAST) as part of Joint Project 2030 (JP2030). The aim of the study was to produce a useable model of the information flows within HQAST and between HQAST and the organisations with which it interacts. The study produced a number of hard-copy charts that record <u>all</u> the information flows captured during the study. The charts are complex, static representations of the information flows. Difficulties in using the charts arise through the charts' complexity that makes it difficult to find the information flows of interest amongst the other information flows shown. The static nature of the charts means that they cannot be customised to show only those information flows and elements of interest to the user. It is also hard to visualise relationships between charts, for instance, when a role within HQAST appears on several charts. Because the charts are hard-copies, they are also difficult to update and validate.

Figure 5-4 below represents how the same information shown in the Aspect charts can be presented to the user in a topographic view. The main view – HQ Australian Theatre – shows the various areas of Defence where information flow analysis was conducted. The user has selected the Operations Directorate within Headquarters Air Command as their current area of interest and a pre-specified topographic view is automatically generated. A set of elements within the Operations Directorate of Air Command are shown. These elements have been laid out according to a horizontal and vertical layout, grouping elements of certain types in vertical columns. Elements in the far left and right columns are external sources and sinks of information respectively. Circles represent the key information items produced and used in Air Command (e.g.

situation reports, operational instructions, air tasking orders etc.), and the coloured rectangles represent roles within Air Command (e.g. Operations Officer 1 (OPSO-1)).

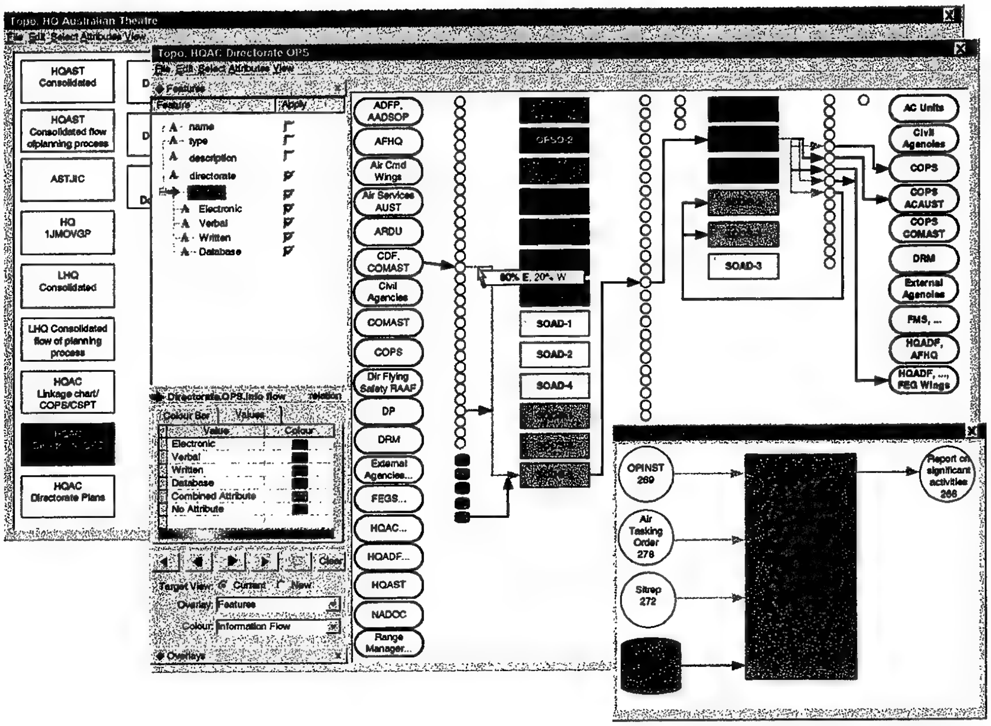

Figure 5-4 - Visualisation of HQAST Information Flows

The Features Overlay Panel allows the selection of various features to be overlaid on the topographic view. For instance, the user may be interested in the flow of information emanating from CDF. In the Features Overlay Panel, the user checks the Info flow box to indicate they wish to see information flows overlaid on the topographic view. The user then selects the types of information flows they wish to see overlaid. In this case the user has indicated that they wish to see whether the information is electronic, verbal, written or from a database. The user then selects the CDF source on the left of the topographic view and clicks the Overlay button on the Features Overlay Panel. The information flows emanating from CDF are displayed as shown in Figure 5-4. Colours of the different types of information flows are specified in the Mapping Table in the Features Overlay Panel.

The user may also be interested in understanding the information flows for specific roles. Figure 5-4 shows a supporting view where the user has selected OPSO-2 to obtain the linked topographic view showing the information flows to and from this role.

This scenario was developed using the information captured by the Aspect study and illustrates how the IV&D Approach may be used to visualise this information in electronic form. Advantages that might be gained from using this approach are that the

problems of complexity could be alleviated by allowing users to customise views by selecting which information flows they wish to see or focusing on particular elements within the system. It may also be possible, through the use of agents, to automate the collection of information and evolution of the Composite Systems Model to ensure descriptions reflect up-to-date information about the system.

6. Research Issues and Directions

The preceding chapters have discussed a range of new concepts and ideas. However they have also raised a significant number of research issues in various areas including visual representation, component-based software engineering, agent support, and domain-based visualisation. The IV&D Approach proposed in this report provides a research framework that can provide the basis for exploring these issues. This chapter looks at these research issues in more detail.

6.1 Visual Representations

The effectiveness of visual representations in providing information which supports specific user needs requires study. There has been considerable work undertaken to define new and novel visual representations. However far less has been done to evaluate how these techniques can be used in practice.

The IVE environment will provide a vehicle for research into visual representations. Its "pluggable" architecture will allow for quick integration of visualisation components. The infrastructure will then allow the visual representations to be studied in terms of their use in particular domains, with the instrumentation and monitoring features supporting empirical studies.

Specific research issues that need study include:

Effectiveness of Visual Representations - What are the criteria relating to effectiveness that need to be considered? How can these be measured/evaluated? Measuring the effectiveness of a system description is dependent on the user of the description and how well the description aids the user in visualising and gaining an understanding of the system. This issue is discussed by Haase (1998) in developing a reference model for "visualisation quality". Robertson and Silver (1995) stress the importance of case studies in "assessing how well particular techniques or approaches have achieved their goals and in focusing what is needed to improve the effectiveness, reliability, and consistency of visualization across a range of fields". Phillips and Vernik (1997) discuss the importance of including instrumentation and analysis features as part of the overall design of computer-based tools. These features then enable researchers to capture usage information that helps in evaluating the effectiveness of the tool for particular tasks and users, usage patterns and user information requirements.

A contributing factor determining the effectiveness of the overall system description is the selection of appropriate representations for primary and supporting views. Lohse

et al. (Lohse, Rueter et al. 1990; Lohse, Biolsi et al. 1994) discuss research undertaken in this area, examining the cognitive structure of eleven types of visual representation and rating each representation in terms of the type of knowledge conveyed, the amount of information conveyed and the ease of use. Further research in this vein will support the IV&D Approach by providing criteria that aids the selection of the most appropriate visual representations for primary and supporting views.

Scalability of Topographic Views - As we are dealing with visualisation of complex systems, which by definition generally consist of a large number of elements and relationships, the issue of how well topographic views cope with scalability is important. For example, topographic views handle the scalability problem in several ways. Firstly, systems may be viewed at different levels of abstraction. One can use different topographic views to view the entire system or only some part of the system. Indeed, a description of the system would be expected to contain a number (or "family") of topographic views. The fact that the views are all based on a single Composite Systems Model ensure that all views are consistent. Secondly, the views can be tailored to present only that information relevant to a user. For example, users have a choice of which elements and relationships they wish to have represented within any given view and a number of alternative views may be used in order for the description to cover the total system. Similarly overlays are specified by users to show only those features that are of interest for that particular view. Thirdly, layout rules can be used to generate topographic views that are a compact spatial representation of the system, i.e. the rules are used to integrate several types of visual representations into a composite form that allows users to view multitudes of system elements of interest and hence help to set the context for investigations. Attribute and relationships information can then be overlaid onto the system representation and viewed in the context of the entire system.

Scope – This issue looks at the range of applicability of the IV&D Approach to different problem domains. It is related to the issue of component generation discussed below. By generating and plugging new components into the IVE Framework one can generate new types of views and increase the scope of applications to which the IVE may be applied. Collection agents may be added to extract new types of information from the information space also increasing the scope of the IVE.

Use of multiple layout and view combinations – The specification of topographic views makes use of a number of methods for laying out elements and relationships at several levels within the one view. An interesting research issue is to investigate the number of possible layout and view structure combinations available, what new topographic patterns and features emerge from these combinations and how may they be best utilised to highlight relevant system information.

Use of novel visualisation techniques – New visualisation techniques are being developed all the time, each having specific advantages for different visualisation problems. Techniques such as 3D visualisation, distortion, and transparency are just a few examples of visualisation techniques available for investigation. The component-based

nature of the IVE will make it possible to integrate new visualisation techniques into the environment as they become available for evaluation.

6.2 Component-Based Software Engineering

Component-based software engineering (CBSE) is an emerging area which is focusing on the rapid assembly of software solutions based on a set of software assets (e.g. off-the-shelf components, frameworks, patterns) and component technologies such as JavaBeans, Enterprise JavaBeans, CORBA, and the various Microsoft component technologies. An international workshop on component-based software engineering stated that the goals of CBSE include the ability to

- Embrace opportunities offered by new technologies in software system delivery and deployment;
- Encourage reuse of core functionality across applications;
- Enable flexible upgrade and replacement of system pieces whether developed in-house, supplied by third parties, or purchased off-the-shelf; and
- Encapsulate organisational best practices such that they can be adapted as business conditions change. (Brown and Wallnau 1998)

In light of the importance of the CBSE for future software development, the systems visualisation research being undertaken as part of the VIDECS research programme is studying the use of CBSE approaches for implementing visualisation solutions. The work on the IV&D Approach reported in this document will act as a demonstrator project to show how CBSE concepts can be realised and to evaluate the various CBSE processes and methods. Results will be published to add to the scientific base of knowledge in this area as well as being transitioned to other Defence domains (e.g. C2*) to support the rapid assembly of component-based software solutions.

A range of research issues regarding the design, description and implementation of component-based systems are being investigated as part of this research project. These include

- Processes for component-based development which focus on the key role of architecture;
- Development of a representation/modelling approach – how do you describe a component system and consistently describe the architectural elements of the design?
- Managing evolution – how do we ensure domain knowledge, design and implementation can evolve smoothly?
- Tool support and implementation technologies – tool support is crucial in order to manage large amounts of integrated domain knowledge, architecture, design and implementation information;

- Use of intelligent components (e.g. agents) to support the deployment of system elements into specific application domains. (Phillips and Vernik 1998)

6.3 Agent support

Agents are components that are used in the IV&D Approach to support the deployment of an IVE into particular application domains through the provision, facilitation and mediation processes. Future work will investigate the infrastructure required to support the use of agents, defining the structure and capabilities of agents, how agents communicate with other agents and the IVE, and whether agents will act individually or have mechanisms for collaborating with other agents. Investigation will be done to evaluate off-the-shelf agent frameworks currently available to see whether they provide a suitable infrastructure for IV&D Agent development.

6.3.1 Characterisation of IV&D Agents

Research must be undertaken to define methods for eliciting the knowledge required by the IV&D Agents to function. Exploration Agents need to map the information space, recognise file types, databases and tools, recognise changes in information items and the structure of the information space, and identify when these changes are important and require update of the CSM. Exploration Agents will also be mobile which means that they must be able to transport themselves across the network in order to monitor the system. The capabilities of Collection Agents are likely to differ for each Collection Agent, depending on the type of information they are able to extract from information items. Work will be carried out on parse agents and fact extractors that elicit information from various information sources (e.g. tools, databases, files). Like the Exploration Agents, Collection Agents may require mobility in order to extract information from information items distributed across a network. This raises issues about how much knowledge the Collection Agents must have, as there is trade-off between amount of knowledge the agent has and its mobility. The more knowledge an agent has the more information needs to be sent over the network when the agent transports itself to another location. Evaluation of current mobile agent development tools and agent frameworks will be conducted to identify appropriate tools that may be exploited by the IVE. Examples of mobile agent development tools are IBM's Aglet Software Development Kit (IBM TRL 1998) and Mitsubishi's Concordia (Mitsubishi Electric 1998). Agent frameworks of interest are Agent Oriented Software's JACK (Agent Oriented Software 1998) and Bits & Pixels Intelligent Agent Library (Bits & Pixels 1998).

Facilitation Agents will support the generation of customised integrated visualisations and CSM evolution based on user needs. The task of Facilitation Agents is to elicit information from the user about the task they are performing and the information required to perform this task. They also support the process of customisation and are able to automate the specification of topographic views based on task information elicited from the user. The knowledge required to do this will be dependent on research discussed above into the use of visual representations and their suitability for

conveying specific types of information to users. Mediation Agents will also require the ability to elicit knowledge from users to obtain user preferences. More importantly, Mediation Agents will require knowledge of the underlying CSM, the visual representations available in the IVE, and how the visual representations may be adapted to convey the information needed by the user. Given that some views may be provided by off-the-shelf components or other components developed for a particular domain, an important issue is how Mediation Agents learn about new visualisation components added to the IVE. Should each visualisation component be added to the IVE along with a Mediation Agent that has knowledge about the component or do visualisation components come with a knowledge base that may be accessed by Mediation Agents when the component is registered with the IVE?

6.3.2 Agent Interaction

How the agents interact with one another is another important issue. Should they have a common pool of knowledge like a blackboard system or interact directly. Facilitation Agents might need the ability to access the CSM in order to determine whether information required by the user is available. If the information is not available but can be obtained by Collection Agents then there is a need for a mechanism to "hire" a Collection Agent to obtain this information. Another example of a need for agent communication is when an Exploration Agent identifies changes in an information item that will have an effect on the CSM. Knowledge about whether a change will effect the model must be conveyed either by giving the Exploration Agent access to the model, or perhaps by some other agent responsible for the model (for instance a Collection Agent) alerting the Exploration Agent about information items that should be monitored for changes.

6.3.3 IVE Deployment Support

As discussed in section 4.5.2, agent components assist in the deployment of IVEs into the user domain. Exploration Agents map the location of information items that describe the system of interest. Collection Agents extract specific attribute and relationship information of system elements from these information items and integrate them into a Composite Systems Model. Facilitation Agents use knowledge of visual representations, user and task information needs to customise visual descriptions for users. Mediation Agents further support users by helping them to adapt descriptions to suit their specific needs and preferences, to explain and highlight important features of visual descriptions and provide explanation of the use of different visual representations.

However, before the agents can be used to automatically deploy an IVE into a user domain, it is necessary to undertake a characterisation study of the domain in order to fully understand the processes and users that must be supported, the tasks the users perform, their information requirements, and the environment in which these tasks are performed (i.e. hardware, software, documents, databases, etc.). For example, before mapping out the location of individual information items, an Exploration Agent will

need to know the location of the physical information resources (e.g. a server) and their network addresses. Information about general user needs may be needed to help Facilitation Agents and Mediation Agents support the customisation and adaptation of visual descriptions. A domain characterisation study is also necessary for the customisation of an IVE as it will identify the necessary IVE components. It will help to specify a system meta-model – a model that represents the different classes of system elements and the relationships that exist between them that are of interest to users. Finally it will identify which information items in the information space describe system elements and relationships, how they are described, and just as importantly, where this information is lacking.

Current research within the Software Systems Engineering Group is working towards the characterisation of the C3I domain (Yates and Vernik et al. 1998). It is envisioned that results from this research will play an important role in providing the knowledge needed to "seed" an IVE for deployment into a C3I domain.

6.4 Domain-Based Visualisation

An important aspect of this research is to study how the IV&D Approach might be applied in different domains. A series of case studies will be conducted to investigate these aspects. There are many examples of where the IV&D Approach could be applied. Some possible case studies are identified in the following paragraphs.

6.4.1 Visualisation of Heterogeneous Software Systems

As has been shown in the visualisation scenarios, the topographic views concept has already been considered in terms of the visualisation of heterogeneous software systems through looking at the JCSE software system. A case study based on a large heterogeneous Command Support System will be conducted once the IVE prototype has been developed. As part of this work, research is currently underway on the characterisation and visualisation of large Lotus Notes C2* systems being used by JCSE and the Air Command Support System (ACSS).

6.4.2 Visualisation of C2* Systems-of-Systems

The IV&D Approach is seen as an important approach for the visualisation of complex C2* systems-of-systems. Results from research into systems characterisation and modelling of C2* systems currently being undertaken within the Software Systems Engineering Group will be used to generate high level visualisations of C2* capability. Case studies will also be undertaken on EXC3ITE to visualise architectural characteristics and system performance.

6.4.3 Visualisation of Enterprise Architectures

Section 5.2 gave a brief insight into the way the IV&D Approach can aid in the visualisation of enterprise architectures and hence help managers of change. Future

case studies will investigate the use of the approach for visualisation of enterprise architectures in the ADO.

6.4.4 Visualisation of Systems of Military Strategy

The ADF system of military strategy may be viewed as a hierarchy consisting of four levels ranging from military strategies at the top level, followed by military strategic options (MSO) (second level), military response options (MRO) (third level). At the bottom level of the hierarchy are the individual force elements (e.g. air/sea/land craft, infantry) which can be used for particular response options. These force elements have a range of attributes that describe aspects such as availability and readiness. Visualisation of this system is important to aid commanders in understanding the different relationships between strategies, MSOs, MROs and the force elements, and the effects that decisions about force elements have on the capability of the ADF to carry out given strategies.

6.4.5 Visualisation of Multi-Agent Systems

The advent of multi-agent systems promises new visualisation problems. As agent systems and individual agents become more sophisticated these systems will be more readily able to adapt and change their behaviour over time depending on their environment and the changing goals of their agents. It will be important to develop ways that enable users to gain an understanding of the state of the multi-agent system and its evolution. Ndumu and Nwana et al. (1997) discuss an approach to visualisation and control of distributed multi-agent applications. Schroeder (1998) discusses three metrics that may be used to measure "distance" between personal, communication and mobile agents respectively. These metrics are used as a basis for developing visualisations of multi-agent systems.

Multi-agent systems are currently being used for land and air battle simulations within the DSTO. Opportunities exist for investigation of issues relating to the use of the IV&D Approach for the visualisation of these multi-agent systems.

7. Conclusions

The increasing complexity of our military systems and their interrelationships demands the development of new techniques for describing and visualising these systems to aid management, operation, acquisition and development. Current computer-based visualisation approaches have provided only a partial solution to the complexity problem. In addition to the continued difficulties in coping with information overload, their effectiveness has also been crippled by their limited scope, their lack of flexibility in adapting to user needs, their large set up costs, and the lack of an underlying model of the system being described.

This report describes the Integrated Visualisation and Description Approach which is proposed as a solution to the problems described above. The key elements of this approach include:

- A conceptual model of the system description process that focuses on supporting the user through facilitation, mediation and provision subprocesses.
- The use of novel topographic views in conjunction with more familiar supporting views for the description of systems.
- A Composite Systems Model that integrates system information filtered automatically from the domain's information space and provides a basis for consistency and integration between all system views.
- The use of a component-based software engineering methodology to develop domain assets which in turn provide the basis for assembling a family of domain specific visualisation tools called Integrated Visualisation Environments (IVEs).
- The use of software agents as a support mechanism for the deployment of IVEs into their domain of use.

A range of future case studies will be undertaken in different domains to further validate the IV&D Approach and provide empirical data to support future development and deployment.

It must be stressed that the work detailed in this report is part of an integrated research programme. In addition to conducting research into how computer-based visualisation can be used to help gain visibility of our Defence systems, we are also conducting research into new software engineering approaches, visual representations, the use of software agents and systems modelling. Contributions from each of these research fields are being employed in the study of the IV&D Approach. Thus the implementation and application of the IV&D Approach may be considered to be a platform for investigations into research issues in each of these fields. This research framework has been designed to focus DSTO research activities in the areas outlined above and to provide a better basis for collaborative research and development with other research organisations, industry and academia.

8. References

Agent Oriented Software (1998). Agent Oriented Software – JACK Intelligent Agents. [Online accessed 3 December, 1998] URL: http://www.agent-software.com.au/jack.html

Allison, J. S. and S. C. Cook (1997). The New Era in Military Systems Thinking and Practice. Adelaide, University of South Australia.

AT&T (1995). AT&T Interactive Data Visualization.

Bits & Pixels (1998). Bits & Pixels Intelligent Agents. [Online accessed 3 December, 1998] URL: http://www.bitpix.com/

Bradshaw, J. M., Ed. (1997). Software Agents. Cambridge, Massachusetts, AAAI/MIT Press.

Brodlie, K. W., L. A. Carpenter, et al. (1992). Scientific Visualization: Techniques and Applications. Berlin, Springer_Verlag.

Brown, A. W. and K. C. Wallnau (1998). "The Current State of CBSE." IEEE Software (September/October): 37-46.

Davidson, C. (1993). "What your Database Hides Away." New Scientist **Jan 1993**.

Delbridge, A. and J.R.L. Bernard, et al., Eds. (1991). The Macquarie Dictionary (Second Edition). The Macquarie Library Pty. Ltd.

DSTO (1997). DSTO - Takari Program Introduction. [Online accessed 29 September, 1998] URL: http://www.dsto.defence.gov.au/esrl/takari/takexint.html

Duffett, P. L. and R. J. Vernik (1997). Software System Visualisation : Netmap Investigations. Technical Report DSTO-TR-0558, DSTO, Adelaide.

Eick, S. G., J. L. Steffen, et al. (1992). "SeeSoft: A Tool for Visualizing Line Oriented Software Statistics." IEEE Transactions on Software Engineering **18**(11): 957-967.

Haase, H. (1998). Evaluating the Quality of Scientific Visualisations: The Q-VIS Reference Model. Conference on Visual Data Exploration and Analysis V, San Jose, California, USA, IS&T/SPIE.

IBM Tokyo Research Laboratory (1998). Aglets Software Development Kit. [Online accessed 3 December 1998] URL: http://www.trl.ibm.co.jp/aglets/

Imagix (1998). Imagix 4D. [Online accessed 9 September 1998] URL: http://www.imagix.com/products/imagix4d.html

Jeffrey, E. V. (1995). Computer Animation and the Law. H. L. Meyer. http://wings.buffalo.edu/Complaw/CompLawPapers/jeffery.html.

Jennings, N. R. and M. Wooldridge (1996). "Software Agents." IEE Review: pp 17-20.

Kaposi, A. and I. Pyle (1993). "Systems are not only software." Software Engineering Journal **Jan 1993**.

Kozaczynski, W. and G. Booch (1998). "Component-Based Software Engineering: Guest Editors Introduction." IEEE Software(September/October): 34-37.

Lohse, G. L., K. Biolsi, et al. (1994). "A Classification of Visual Representation." Communications of the ACM **37**(12): 36-49.

Lohse, J., H. Rueter, et al. (1990). Classifying Visual Knowledge Representations: A Foundation for Visualization Research. First IEEE Conference on Visualization, San Francisco.

LUSAS (1998). LUSAS Finite Element Analysis Product Options. [Online accessed 9 September 1998] URL: http://www.lusas.com/productoptions.html

Maier, M. W. (1997). "Architecting Principles for Systems-of-Systems." Systems Engineering (in review). [Online accessed, 17 June, 1998] URL: http://www.infoed.com/Open/PAPERS/systems.htm

McCormick, B. H., T. A. DeFanti, et al. (1987). "Visualisation in Scientific Computing." *Computer Graphics (AC-SIGGRAPH)* **21**(6): 1-15.

Mitsubishi Electric America (1998). What is Concordia? [Online accessed 3 December, 1998] URL: http://web.vsisinc.com/concordia/product_information/what_is_it.htm

NASA (1992). NASA Systems Engineering Handbook, Draft.

Ndumu, D. T., H. S. Nwana, et al. (1997). "Advanced Visualisation of Distributed Multi-agent Systems." Submitted for Publication Consideration for CHI '98 Conference.

Nwana, H. (1996). "Software Agents: An Overview." Knowledge Engineering Review **11**(3): 205-244.

Owens, W. A., Ed. (1997). The Emerging U.S. System of Systems. Dominant Battlespace Knowledge, NDU. [Online accessed 17 June, 1998]. URL: http://www.ndu.edu/ndu/inss/books/dbk/dbkch01.htm

Phillips, M. and R. Vernik (1998). Defining and Developing Component System Architectures. Component-Oriented Software Engineering Workshop, ASWEC 98, Adelaide.

Phillips, M. P. and R.J. Vernik (1997). Capturing and Analysing Usage of Interactive Computer-Based Tools. Research Report DSTO-RR-0119, DSTO, Adelaide.

Price, B. A., R. M. Baecker, et al. (1993). "A Principled Taxonomy of Software Visualisation." Journal of Visual Languages and Computing **4**(3): 211-266.

Rational Software Corporation (1998). Rational Rose 98.

Robertson, P. K. and D. Silver (1995). "Visualization Case Studies: Completing the Loop." IEEE Computer Graphics and Applications **15**(4): 18-19.

Rogers, E. (1995). "Cognitive Cooperation through Visual Interaction." Knowledge-Based Systems **8**(2): 117-125.

Schroeder, M. (1998). Towards Visualisation of Multi-Agent Systems. Workshop on Interaction Agents, Aquila, Italy.

Shneiderman, B. (1983). "Direct Manipulation: A step beyond programming languages." IEEE Computer **16**(8): 57-68.

Szyperski, C. A. (1997). Component Software: Beyond Object-Oriented Programming, Addison-Wesley.

USAF Human Systems Center of Air Force Materiel Command (1998). Personal Computer Software System for Crewmember Ejection and Crash Analysis. [Online accessed 3 December, 1998] URL: http://www.brooks.af.mil/HSC/products/doc36.html

US DoD (1998). DoD Dictionary of Military Terms. **1998.**
http://www.dtic.mil/doctrine/jel/doddict/

Vernik, R. J. (1996). Visualisation and Description in Software Engineering. Ph.D. Thesis. Computer and Information Science. Adelaide, University of South Australia: 232.

Yates, A. J., R. Vernik, et al. (1998). Systems Characterisation and Modelling Approaches for C3I. DSTO, Adelaide.

Integrated Visualisation and Description of Complex Systems

D.P.J. Goodburn, R.J. Vernik, M.P. Phillips and J.J. Sabine

(DSTO-RR-0154)

DISTRIBUTION LIST

	Number of Copies
AUSTRALIA	
DEFENCE ORGANISATION	
Task sponsor:	
DGCSS	1
S&T Program	
Chief Defence Scientist)	
FAS Science Policy)	1 shared copy
AS Science Corporate Management)	
Director General Science Policy Development	1
Counsellor, Defence Science, London	Doc Control Sheet
Counsellor, Defence Science, Washington	Doc Control Sheet
Scientific Adviser - Policy and Command	1
Navy Scientific Adviser	1 copy of Doc Control Sheet and 1 distribution list
Scientific Adviser - Army	Doc Control Sheet and 1 distribution list
Air Force Scientific Adviser	1
Director Trials	1
Aeronautical & Maritime Research Laboratory	
Director	1
Electronics and Surveillance Research Laboratory	
Director	1
Chief Information Technology Division	1
Research Leader Command & Control and Intelligence Systems	1
Research Leader Military Computing Systems	1
Research Leader Command, Control and Communications	1
Executive Officer, Information Technology Division	Doc Control Sheet
Head, Information Architectures Group	1

Head, Information Warfare Studies Group	Doc Control Sheet
Head, Software Systems Engineering Group	Doc Control Sheet
Head, Year 2000 Project	Doc Control Sheet
Head, Trusted Computer Systems Group	Doc Control Sheet
Head, Advanced Computer Capabilities Group	1
Head, Systems Simulation and Assessment Group	Doc Control Sheet
Head, C3I Operational Analysis Group	Doc Control Sheet
Head Information Management and Fusion Group	1
Head, Human Systems Integration Group	Doc Control Sheet
Head, C2 Australian Theatre	1
Head, Information Architectures Group	1
Head, Distributed Systems Group	Doc Control Sheet
Head C3I Systems Concepts Group	1
Head, Organisational Change Group	Doc Control Sheet
Dr Daniel Goodburn, SSE Group ITD	1
Mr Matthew Phillips, SSE Group ITD	1
Mr Justin Sabine, SSE Group ITD	1
Publications and Publicity Officer, ITD/Executive Officer, ITD	1
DSTO Library and Archives	
Library Fishermans Bend	1
Library Maribyrnong	1
Library Salisbury	2
Australian Archives	1
Library, MOD, Pyrmont	Doc Control Sheet
Capability Development Division	
Director General Maritime Development	Doc Control Sheet
Director General Land Development	Doc Control Sheet
Director General C3I Development	Doc Control Sheet
Director General Aerospace Development	Doc Control Sheet
Navy	
SO (Science), Director of Naval Warfare, Maritime Headquarters Annex, Garden Island, NSW 2000.	Doc Control Sheet
Army	
ABCA Standardisation Officer, Puckapunyal	4
SO (Science), DJFHQ(L), MILPO, Enoggera, Qld 4051	Doc Control Sheet
NAPOC QWG Engineer NBCD c/- DENGRS-A, HQ Engineer Centre, Liverpool Military Area, NSW 2174	Doc Control Sheet
Intelligence Program	
DGSTA Defence Intelligence Organisation	1

Corporate Support Program (libraries)	
OIC TRS Defence Regional Library, Canberra	1
US Defence Technical Information Center,	2
UK Defence Research Information Centre,	2
Canada Defence Scientific Information Service,	1
NZ Defence Information Centre,	1
National Library of Australia,	1
Universities and Colleges	
Australian Defence Force Academy	1
Library	1
Head of Aerospace and Mechanical Engineering	1
Deakin University, Serials Section (M list)), Deakin University Library, Geelong, 3217	1
Senior Librarian, Hargrave Library, Monash University	1
Librarian, Flinders University	1
Other Organisations	
NASA (Canberra)	1
AGPS	1
State Library of South Australia	1
Parliamentary Library, South Australia	1

OUTSIDE AUSTRALIA

Abstracting and Information Organisations	
Library, Chemical Abstracts Reference Service	1
Engineering Societies Library, US	1
Materials Information, Cambridge Scientific Abstracts US	1
Documents Librarian, The Center for Research Libraries, US	1
Information Exchange Agreement Partners	
Acquisitions Unit, Science Reference and Information Service, UK	1
Library - Exchange Desk, National Institute of Standards and Technology, US	1
National Aerospace Laboratory, Japan	1
National Aerospace Laboratory, Netherlands	1
SPARES	10
Total number of copies:	**68**

Page classification: UNCLASSIFIED

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
DOCUMENT CONTROL DATA

1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)
N/A

2. TITLE	3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)
Integrated Visualisation and Description of Complex Systems	Document (U) Title (U) Abstract (U)
4. AUTHOR(S)	5. CORPORATE AUTHOR
D.P.J. Goodburn, R.J. Vernik, M.P. Phillips and J.J. Sabine	Electronics and Surveillance Research Laboratory PO Box 1500 Salisbury SA 5108

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-RR-0154	AR-010-993	Research Report	June 1999

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
N9505/17/94	DAO 97/127	DGCSS	57	42

13. DOWNGRADING/DELIMITING INSTRUCTIONS	14. RELEASE AUTHORITY
N/A	Chief, Information Technology Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE CENTRE, DIS NETWORK OFFICE, DEPT OF DEFENCE, CAMPBELL PARK OFFICES, CANBERRA ACT 2600

16. DELIBERATE ANNOUNCEMENT
No limitations

17. CASUAL ANNOUNCEMENT No

18. DEFTEST DESCRIPTORS
Visualisation
Complex systems
Software engineering
Software tools

19. ABSTRACT
This report discusses the Integrated Visualisation and Description (IV&D) Approach, a computer-based visualisation approach that is being developed to support the visualisation and description of complex systems. Guided by a conceptual model of a description process that is driven by user information needs within a domain context, the approach incorporates the use of novel visualisation techniques based on system topographies and feature overlays. System information from the domain's information space is filtered and integrated into a Composite Systems Model that provides a basis for consistency and integration between all system views. A component-based software engineering methodology is described for the development of domain assets which provide the basis for developing a family of domain specific visualisation tools, called Integrated Visualisation Environments (IVEs), and deploying IVEs in their domain of use through the support of software agents. Examples of use of the IV&D Approach for the visualisation of heterogeneous software systems, C2* systems-of-systems and enterprise architectures are discussed, as are future research issues to be investigated.

Page classification: UNCLASSIFIED